TECHNISCHE UNIVERSITÄT CHEMNITZ

Fakultät für Elektrotechnik und Informationstechnik

Professur für Mikrosystem- und Gerätetechnik

Diplomarbeit

Algorithmische Optimierung von Teststrukturen zur Charakterisierung von Mikrosystemen auf Waferebene

Petra Streit

Chemnitz, den 4. April 2009

Prüfer: Prof. Dr.-Ing. habil. Jan Mehner
Betreuer: Dipl.-Ing. Alexey Shaporin
Dipl.-Ing. Marco Dienel

Streit, Petra
Algorithmische Optimierung von Teststrukturen zur Charakterisierung von Mikrosystemen auf Waferebene
Diplomarbeit, Professur für Mikrosystem- und Gerätetechnik
Technische Universität Chemnitz, April 2009

Kurzzusammenfassung

Diese Diplomarbeit beschäftigt sich mit der Entwicklung von Teststrukturen zur Charakterisierung von Mikrosystemen auf Waferebene. Sie dienen zur Bestimmung von Prozesstoleranzen. Ziel dieser Arbeit ist es, einen Algorithmus zu entwickeln, mit dem Teststrukturen optimiert werden können. Dazu wird ein Ansatz zur Optimierung von Teststrukturen mittels eines Genetischen Algorithmus untersucht. Grundlage für diesen ist eine Bewertung der Strukturen hinsichtlich der Sensitivität gegenüber den Fertigungsparametern und der Messbarkeit der Eigenmoden. Dem Leser wird zuerst ein Einblick in das Themengebiet und in die Verwendung von Teststrukturen gegeben. Es folgen Grundlagen zur Fertigung und Messung von Mikrosystemen, zur Parameteridentifikation, sowie zu Optimierungsalgorithmen. Anschließend wird ein Bewertungs- und Optimierungskonzept, sowie eine Softwareimplementation für die sich aus der Optmierung ergebenden Aufgaben, vorgestellt. Unter anderem eine Eigenmodenerkennung mittels Neuronalem Netz und einer auf der Vandermond'schen Matrix basierende Datenregression. Die Ergebnisse aus der Umsetzung durch ein Testframework werden abschließend erläutert. Es wird gezeigt, dass die Optimierung von Teststrukturen mittels Genetischem Algorithmus möglich ist. Die dargestellte Bewertung liefert für die untersuchten Teststrukturen nachvollziehbare Resultate. Sie ist in der vorliegenden Form allerdings auf Grund zu grober Differenzierung nicht für den Genetischen Algorithmus geeignet. Entsprechende Verbesserungsmöglichkeiten werden gegeben.

Abstract

This diploma thesis deals with the development of test-structures for the characterization of microsystems on wafer level. Test-structures are used for the determination of geometrical parameters and material properties deviations which are influenced by microsystem fabrication prozesses. The aim of this work is to establish principles for the optimization of the test-structures. A genetic algorithm as an approach for optimization is investigated in detail. The reader will get an insight in the topic and the application of test-structures. Fundamentals of fabrication and measurement methods of microsystems, the parameter identification procedure and algorithms for optimization follow. The procedures and a corresponding software implementation of some applied issues, which are needed for the optimization of test structures, are presented. Among them are neural network algorithms for mode identification and a data regression algorithm, based on Vandermonde Matrix. Results of implemented software algorithms and an outlook conclude this work. It is shown, that the optimization of test-structure using a genetic algorithm is possible. An automated parameter variation procedure and the extraction of important test-structures parameters like sensitivity and mode order are working properly. However, the presented evaluation is not suitable for the genetic algorithm in the presented form. Hence, improvements of evaluation procedure are suggested.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

APDL	ANSYS Parametric Design Language
BPN	Backpropagation Netzwerk
BDRIE	Bonding and Deep Reactive Ion Etching
CMake	Cross-plattform Make
DRIE	Deep Reactive Ion Etching
EA	Evolutionärer Algorithmus
ES	Evolutionsstrategien
FANN	Fast Artificial Neural Network
FE	Finite Elemente
FEM	Finite Elemente Methode
GA	Genetischer Algorithmus
GUI	Graphical User Interface
LDV	Laser Doppler Vibrometer
MEMS	Mikro-Elektro-Mechanisches Systemes
MIT	Massachusetts Institute of Technology
NN	Neuronales Netz
PAP	Programmablaufplan
SMD	Spring-Mass-Damping
TMDE	Time Multiplexed Deep Etching
TS	Teststruktur
TTV	Total Thickness Variation

Kapitel 1

Einführung

Geometrie- und Materialparameter bestimmen das Verhalten Mikro-Elektro-Mechanischer Systeme (MEMS). Diese Parameter sind beispielsweise Strukturdicke, Unterätzung und intrinsischer mechanischer Stress. Mit der steigenden Komplexität und fortschreitenden Miniaturisierung von Mikrosystemen stellt sich die Aufgabe der Entwicklung von geeigneten Charakterisierungsmöglichkeiten auf Waferebene. Der Prozess soll aus Zeit- und Kostengründen während des Herstellungsprozesses erfolgen.

In diesem Kapitel wird eine kurze Übersicht über das Themengebiet gegeben und die Gründe für die Verwendung spezieller Teststrukturen erläutert. Daraus ergibt sich die Motivation dieser Arbeit. Der Stand der Technik wird im anschließenden Abschnitt umrissen und abschließend werden einige Einsatzmöglichkeiten verwendeter Optimierungsverfahren in Verbindung mit MEMS aufgezeigt.

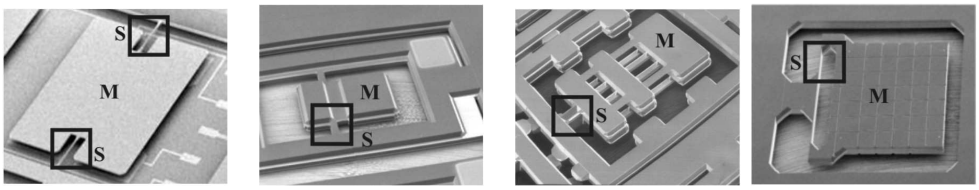
Im folgenden Kapitel werden einige wichtige Grundlagen zusammengefasst. Kapitel 3 verdeutlicht das Konzept der Arbeit und Kapitel 4 die entsprechende Umsetzung, gefolgt von den Ergebnissen. Das Kapitel 6 zieht ein Fazit und gibt einen Ausblick zur Fortsetzung des Themas.

1.1 Hinleitung zum Thema

Mikromechanische Sensoren und Aktuatoren, wie beispielsweise Beschleunigungssensoren, Mikrospiegel und Mikrospiegelarrays, bilden eine wichtige Gruppe von MEMS, die Feder-Masse-Dämpfer Systeme (engl. Spring-Mass-Damping (SMD)). Ihre Beschreibung erfolgt durch ihr Systemverhalten, welches von Fertigungsparametern abhängig ist. Abbildung 1.1[1] zeigt einige dieser SMD Systeme. Die Feder-, Masse- und Dämpferelemente sind gut zu unterscheiden. Typischerweise verbinden Federelemente (`S`) Massenelemente (`M`) mit einem festen Rahmen. Die Dämpfung erfolgt durch die umgebenden Luft.

Die Informationen über technologische Prozesstoleranzen und Materialeigenschaften (wie in Abbildung 1.2 angedeutet) sollen innerhalb der Prozesskette zer-

[1] Quelle: Technische Universität Chemnitz, Zentrum für Mikrotechnologien (ZfM)



(a) Optisches System (b) Drucksensor (c) Beschleunigungssensoren

Abbildung 1.1: Beispiele für SMD-Systeme

störungsfrei für den ganzen Wafer ermittelt werden. Um den Prozessablauf nicht zu behindern, muss der Vorgang in wenigen Minuten oder sogar Sekunden durchgeführt werden. Die gewonnen Daten werden für die Überwachung des Fertigungsprozesses genutzt.

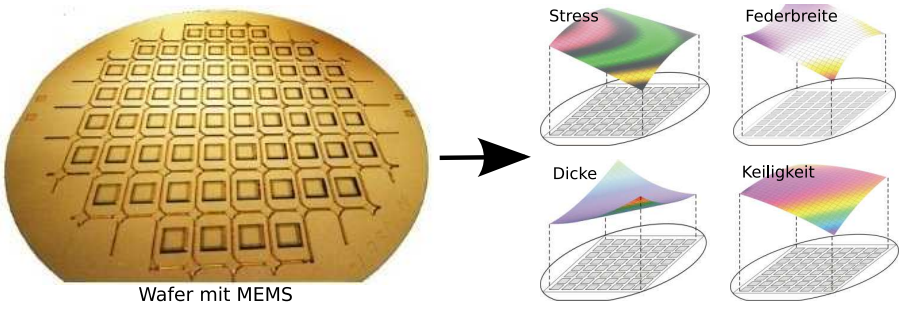


Abbildung 1.2: Verteilung von Prozesstoleranzen über Wafer

Viele dieser Parameter sind, wie in [SFS+08] erläutert, durch direkte Messmethoden nicht ermittelbar. Daher muss eine indirekte Methode genutzt werden. Für die Charakterisierung von MEMS werden statt der eigentlichen Strukturen spezielle kleinere Teststrukturen verwendet. Die Gründe für deren Verwendung werden im nachfolgenden Abschnitt 1.2 beschrieben.

Kontaktfreie Messtechniken sind bei Anregung und Antwortsignaldetektion zu bevorzugen. Es werden dynamische Charakteristiken, wie Eigenfrequenzen und die zugehörigen Eigenmoden, verwendet. Diese sind unabhängig von der Anregungsamplitude und geben Informationen über das mechanische Verhalten der betrachteten Komponenten.[SFS+08]

Die verwendete Messtechnik spiegelt sich in der gesamten Parameteridentifikation wider. Sie besteht im Wesentlichen aus drei Schritten. Im ersten Schritt wird die theoretische Anwort der Teststruktur durch numerische Modalanalyse ermittelt. Anschließend erfolgt die Messung der Antwort der realen Teststruktur. Daraufhin werden die theoretischen Daten mit den gemessenen in Verbindung gesetzt und die gesuchten Parameter ermittelt. Abbildung 1.3 zeigt zusammenfassend den kompletten Ablauf der Charakterisierung von Material- und Fertigungsparametern. Die Teststrukturen werden zusammen mit MEMS auf einem Wafer gefertigt. Aus der Parameteridentifikation werden Karten der gesuchten Parameter für den gesamten

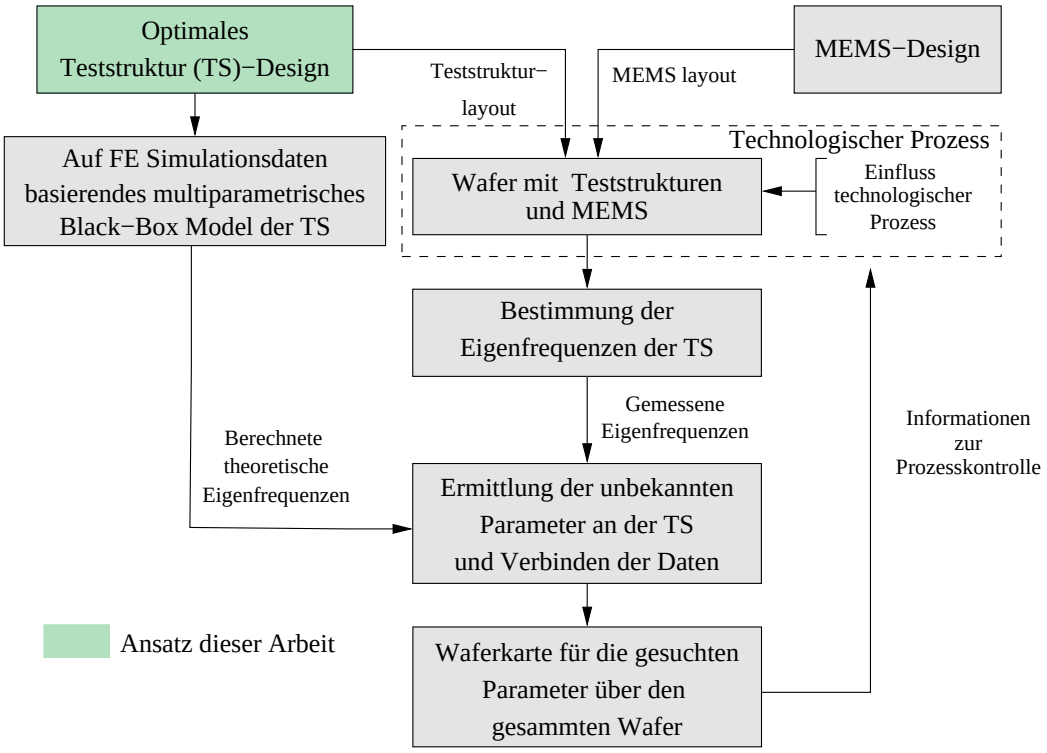


Abbildung 1.3: Ablauf der Charakterisierung von Waferparametern (nach [SHF+07])

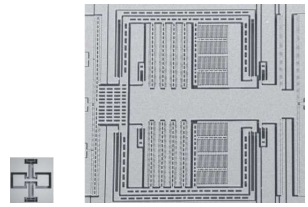
Wafer erzeugt. Aus diesen können Informationen zur Prozesskontrolle gewonnen werden.

1.2 Verwendung von Teststrukturen

Bei der Fertigung von MEMS treten, wie eingangs erläutert, produktionsbedingte Toleranzen auf. Die entsprechenden Parameter müssen identifiziert werden. Dazu ist ein effizientes Verfahren notwendig. Basis für dieses sind spezielle Teststrukturen, die nur der Identifizierung von Fertigungsparametern dienen.

Die Teststrukturen sind um ein Vielfaches kleiner als reale MEMS-Strukturen, wie Abbildung 1.4 deutlich macht. Dadurch können sie ressourcensparend auf dem Wafer im Raum zwischen den MEMS platziert werden, in welchem diese im letzten Fertigungsschritt durch Sägen getrennt werden (siehe Abbildung 1.5).

Die Vorteile bei der Verwendung von Teststrukturen liegen in der Zeitersparnis, da nur eine einmalige Entwicklung und Simulation der Teststrukturen erfolgen muss. Diese Simulation von, im Vergleich zum MEMS einfachen, Strukturen benötigt weniger Zeit. Weiterhin besteht die Wiederverwendbarkeit in Verbindung mit verschiedenen MEMS. Sie erlauben standardisierte Testmethoden und können hinsichtlich einer erhöhten Sensitivität gegenüber spezifischen Prozessparametern ausgelegt werden.



(a) TS (b) MEMS

Abbildung 1.4: Größenvergleich zwischen Teststruktur und realer MEMS-Struktur

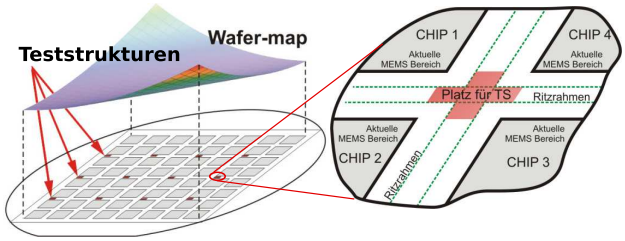


Abbildung 1.5: Platzierung von Teststrukturen auf einem Wafer

Die Anforderungen an eine TS umfassen unter anderem eine maximale Strukturgröße, wegen der Platzierung auf dem Wafer, eine Mindestfläche für die Messung und eine klare Erkennbarkeit der Eigenmoden. Die Mindestanzahl dieser unabhängigen Eigenmoden ist durch die Anzahl der unbekannten Parameter gegeben (siehe Abschnitt 2.5). Eigenmoden in Out-of-Plane-Richtung sind auf Grund der besseren Messbarkeit zu bevorzugen. Die Sensitivität gegenüber der zu identifizierenden Parameter soll hoch sein.

Die Teststrukturen sind ebenfalls Feder-Masse-Dämpfungssysteme, wobei die Dämpfung klein sein soll. Die Breiten der Federn liegen im für MEMS typischen Bereich von 3 bis 10 μm.

Die Herausforderung bei der Entwicklung von Teststrukturen besteht darin, eine „gute“ Struktur zu finden, welche die vorgegebenen Anforderungen erfüllt. Eine Bewertung jeder einzelnen ist zur Vergleichbarkeit untereinander notwendig. Ein Konzept zur Optimierung von Teststrukturen wird in Abschnitt 3.3 vorgestellt.

Im Abschnitt 2.5 wird der Ablauf der Parameteridentifikation mit Hilfe von Teststrukturen kurz erläutert.

1.3 Motivation dieser Arbeit

Die vorliegende Arbeit beschäftigt sich mit den für die Charakterisierung von MEMS benötigten Teststrukturen. Diese müssen bewertet und hinsichtlich ihrer Form und der Sensitivität gegenüber der gesuchten Parameter optimiert werden. Je empfindlicher eine Teststruktur bezüglich eines Parameters ist, desto sicherer lässt sich dieser identifizieren. Die für die Parameteridentifikation benötigten Eigenmoden müssen sich zudem eindeutig zuordnen lassen und die Eigenfrequenzen damit innerhalb des messbaren Bereiches liegen. Zusätzlich dürfen sie auf Grund ihrer Positionierung auf dem Wafer eine bestimmte Größe nicht überschreiten.

Für diese Optimierungsaufgabe gibt es zahlreiche Ansätze. Die Wahl fiel dabei anfänglich auf Evolutionäre Algorithmen, Neuronale Netze und Zelluläre Automaten. Einige Anwendungen dieser in Verbindung mit MEMS werden in Abschnitt 1.5 zusammengefasst. Warum kein klassisches Optimierungsverfahren gewählt wurde,

wird in Abschnitt 2.6 verdeutlich.

Nach eingehender Untersuchung der Leistungsfähigkeit und der Möglichkeiten dieser Algorithmen zeigte sich, dass Zelluläre Automaten für diese Arbeit keine Relevanz besitzen. Die Begründung dazu findet sich in Abschnitt 1.5. Evolutionäre Algorithmen scheinen geeignet, um die Teststrukturen in ihrer geometrischen Erscheinung zu optimieren. Dies wurde bereits für andere MEMS umgesetzt (siehe Abschnitt 1.5). Neuronale Netze leisten keinen direkten Beitrag zur Erstellung neuer Strukturen. Jedoch ist im Rahmen der Simulation eine Eigenmodenerkennung notwendig, welche als assoziative Aufgabe in ihren Anwendungsbereich fällt.

Der derzeitige Stand der Technik in der Verwendung von Teststrukturen, ebenso wie der Ansatzpunkt dieser Arbeit wird in Abschnitt 1.4 dargelegt.

In Kapitel 3 wird das Konzept der Optimierung von Teststrukturen schrittweise erläutert, Kapitel 4 behandelt die Umsetzung. Die Ergebnisse folgen in Kapitel 5.

1.4 Stand der Technik

Zur indirekten Charakterisierung von MEMS existieren bereits einige Verfahren. Gupta [GOS96] stellte 1996 den M-Test vor. Der M-Test basiert auf der Messung des elektrostatischen Pull-In eines Sets von einfachen Teststrukturen in polykristallinem Silizium. [OS97] Diese Methode benötigt allerdings eine elektrische Kontaktierung zu dem zu testenden Wafer.

An der Technischen Universität Chemnitz wurde 1997 eine Technik zur experimentellen Anpassung für Modellparameter auf Basis von Daten des Finite Elemente (FE)-Modells und gemessener Eigenfrequenzen eines MEMS entwickelt.

Gupta [Gup00] präsentierte 2000 eine Messtechnik zur Ermittlung geometrischer Parameter durch Messung und Simulation einer Mikrostruktur auf Waferebene. Die Basis bildeten analytisch berechnete und gemessenene Eigenfrequenzen.

Die Basis für die in dieser Arbeit beschriebene Parameteridentifikation legten 2005 Shaporin et al. [SHD05] mit der Charakterisierung der Geometrie und des Stresses in einer Mikrospiegelanordnung aus einkristallinem Silizium. Dafür wurde das MEMS selbst und keine Teststruktur verwendet. Teststrukturen erweiterten später diese Technik.[SHF+07]

Die erste Generation von Teststrukturen hat ein einfaches Layout (siehe Abbildung 1.6). Sie umfassen Strukturen mit mechanischer Stresskompensation zur Bestimmung von Dicke, Maskenunterätzung und Winkel der Seitenwände sowie Strukturen ohne Stresskompensation zur Ermittlung von mechanischem Stress. Weiterhin sind im Testfeld einseitig eingespannte Federn mit seismischer Masse für das Elastizitätsmodul und die Dichte des Materials vorhanden. [SFS+08]

Zur Optimierung der Teststrukturen werden die normierten Eigenfrequenzen f_{norm} in Abhängigkeit von den gesuchten Parametern — in Abbildung 1.7 dem Parameter p — gesetzt. Der Winkel α zwischen den Kurven der Eigenfrequenzen quantifiziert die Genauigkeit der Parameteridentifikation. Im optimalen Fall stehen

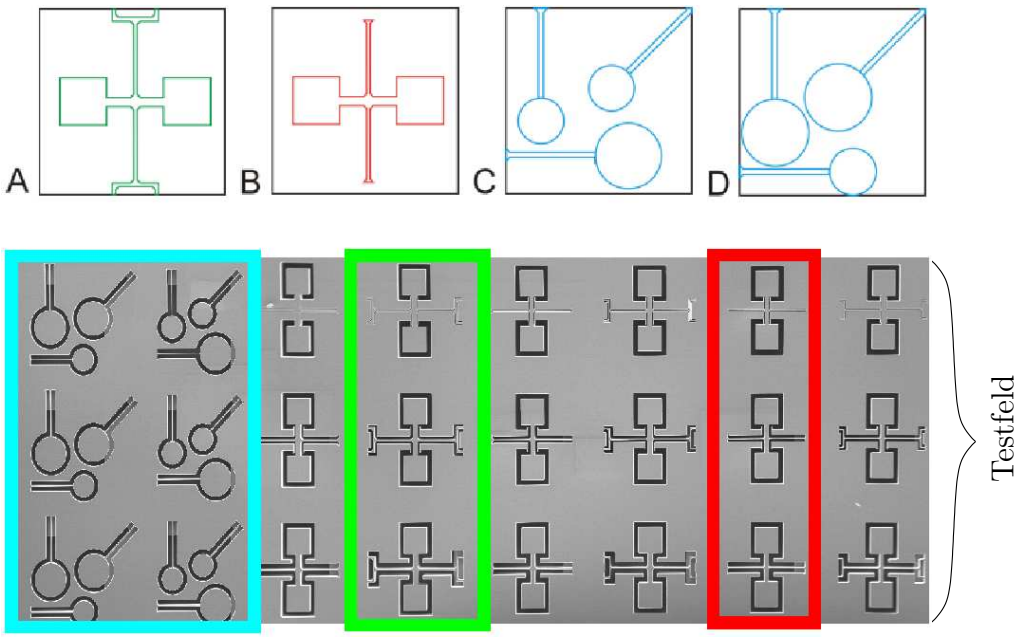
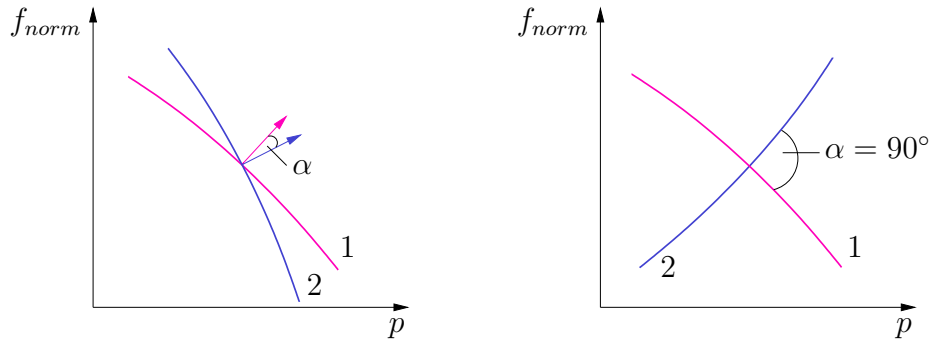


Abbildung 1.6: TS des aktuellen Testfeldes

die Kurven senkrecht aufeinander. Für mehr als zwei Parameter wird die Optimierung komplexer.[SHD05]



(a) Nichtoptimale Abhängigkeit bei zwei Eigenfrequenzen

(b) Optimale Abhängigkeit bei zwei Eigenfrequenzen

Abbildung 1.7: Eigenfrequenzabhängigkeit von einem Parameter

Der Aufwand für den Optimierungsprozess steigt mit der Anzahl der gesuchten Parameter erheblich an. Die Optimierung erfolgt nur an einer einzelnen Teststruktur. Es müssen zahlreiche Strukturen untersucht, um eine geeignete für die Identifizierung bestimmter Parameter zu finden. Auf Grund des enormen Zeit- und Arbeitsaufwandes wird nach Möglichkeiten gesucht, diesen Schritt zu automatisieren und damit eine große Anzahl von Teststrukturen zu untersuchen.

Diese Arbeit legt die Grundlagen für diese Aufgabe und zeigt eine Beispielumsetzung auf.

1.5 Optimierungsverfahren für MEMS

Auf Grund der erwähnten Komplexität des Entwickelns von MEMS müssen eine große Menge Ressourcen investiert werden. Viele dieser Investitionen werden durch den iterativen Trial-and-Error Designprozess (siehe Abschnitt 2.7) verbraucht. Um diesen Entwicklungsschritt zu verbessern, werden verschiedene Optimierungsverfahren genutzt.

Eine Automatisierung hilft den Entwicklern schneller geeignete Randbedingungen zu finden und damit den Entwicklungsprozess für MEMS für eine bestimmte Herstellungstechnologie zu beschleunigen.

Zur Verbesserung bestehender Strukturen gibt es zahlreiche Ansätze. Nicht jedes Optimierungsverfahren wird den Anforderungen gerecht. Aus diesem Grund müssen neben den klassischen Verfahren auch Alternativen untersucht werden.

In diesem Abschnitt werden einige Anwendungsbeispiele für ausgewählte Optimierungsverfahren in der Entwicklung von MEMS beschrieben. Die Auswahl beschränkt sich dabei auf Evolutionäre Algorithmen, Neuronale Netze und Zelluläre Automaten. Diese haben den Vorteil, dass sie für viele Aufgaben angepasst werden können.

Fan et al. [FGW+04] beschreibt die Anwendung eines evolutionären Ansatzes zur Automatisierung der Synthese von MEMS. Ziel ist es, sich wiederholende Aufgaben zu eliminieren und durch parallel arbeitende Evolutionäre Algorithmen bessere Lösungen zu finden. Die Entwicklung eines MEMS wird dafür in zwei Ebenen geteilt — in das Makromodell für das Systemverhalten und das geometrische Layout. Auf Systemebene wird eine Kombination aus genetischer Programmierung und Verbindungsgraphen genutzt, um passende Designkandidaten für die Spezifikation zu finden. In der Layoutebene werden Genetische Algorithmen zur Optimierung von Geometrieparametern eingesetzt, um das physische Modell an das Verhaltensmodell anzupassen. Die Umsetzung wird an einem Multiresonator-Mikrosystem verdeutlicht.

Zhang et al. [ZKAS05] stellt eine hierarchische MEMS-Synthese und Optimierungsarchitektur vor. Diese enthält eine objektorientierte Komponentenbibliothek mit einem MEMS-Simulationswerkzeug und einer zweistufigen Optimierung. Die Stufen gliedern sich in globale Genetische Algorithmen und lokaler Verfeinerungen mit einem Gradientenverfahren. Die Komponentenbibliothek enthält primitive Elemente und komplexere Anordnungen dieser. Zusätzlich beinhalten sie Verbindungsbedingungen und Randbedingungen für die genetischen Operationen Mutation und Kreuzung. Die Verwendung dieses mehrdimensionalen Genetischen Algorithmus (Multi-Objective Genetic Algorithm MOGA) wird am Beispiel eines Resonators mit Meanderfedern dargestellt. Dieser soll bei einer möglichst kleinen Fläche eine bestimmte Mindeststeifigkeit und eine minimale Abweichung von der gegebenen Resonanzfrequenz aufweisen. Die Ergebnisse zeigen, dass die zweistufige Optimierung den Vorteil hat, Strukturen im Layout automatisch zu variieren und damit eine Vorauswahl möglicher Designvarianten zu ermitteln. Mit dem Gradientenverfahren werden diese weiter verbessert. Es wäre allein nicht in der Lage beispielsweise neue Elemente an einer Struktur zu ergänzen. In [ZAKS06] ist die

auf MOGA basierende Synthese näher beschrieben.

Li und Antonsson [LA98] beschreiben eine automatische Synthese von Maskenlayouts für MEMS mittels Genetischem Algorithmus. Dafür wird aus einer gegebenen Startpopulation zulässiger zweidimensionaler Maskenlayouts jeweils ein 3D-Ätzprofil erstellt und mit dem zu erzielenden verglichen. Über einen Genetischen Algorithmus werden die Maskenlayouts verändert und ein entsprechendes Ätzprofil gesucht, welches dem zu erzielenden möglichst nahe kommt.

Wie deutlich wird, können Evolutionäre Algorithmen genutzt werden, um bestehende Strukturen zu optimieren. Dabei liegt das Hauptaugenmerk oft auf Geometrien, die modifiziert und erweitert werden. Aus bestehenden Bausteinen können komplexere erstellt und weiterverwendet werden. Somit werden auch Strukturen berücksichtigt, die vom Entwickler aus Zeit– oder Komplexitätsgründen nicht in Betracht gezogen werden können. In dieser Arbeit liegt der Schwerpunkt auf dem Verändern vorgegebener Grundgeometrien.

In manchen Fällen ist nicht die Generierung neuer Strukturen, sondern ein Vergleich mit vorgegebenen sinnvoll. Das Gleiche gilt für das Verhalten eines Systemes. Asgary und Mohammadi [AM05] beschreiben die Verwendung Neuronaler Netze als Fehlererkennungsmechanismus für MEMS. Die Reaktionen des Mikrosystemes werden für häufige Fehler wie Ermüdung, Brüchigkeit, Ätzabweichungen, Verunreinigungen und Stickingeffekte ebenso wie für das fehlerfreie Verhalten simuliert. Diese werden dem Neuronalen Netz als Muster zum Training vorgelegt. Auf Basis dessen klassifiziert das Neuronale Netz Eingabemuster und entscheidet, ob das Mikrosystem fehlerbehaftet oder fehlerfrei ist. In diesem Fall wird das Neuronale Netz zur Mustererkennung eingesetzt.

Litovski et al. [LAZ04] beschreibt ein ähnliches Verfahren anhand eines mikroelektro-mechanischen kapazitiven Drucksensors. Eine erstellte Liste von Fehlerkodes und zugehörigen Fehlverhalten wird dem Neuronalen Netz gelernt, welches anschließend in der Diagnose eines Systemes den wahrscheinlichsten Fehlerkode liefert. Der Vorteil Neuronaler Netze, auch bei verfälschten Eingaben ein korrektes Ergebnis zu liefern, kommt hier zum Tragen.

Ilumoka und Tan [IT07] beschreiben die Verwendung Neuronaler Netze zur Vorhersage von Versagenswahrscheinlichkeiten von MEMS und der Qualitätsverbesserung. Dazu stellen sie Eigenschaften von MEMS Lebenszeitstatistiken gegenüber. Das Neuronale Netz sagt dann für ein Mikrosystem mit bestimmten Eigenschaften die mittlere Lebenszeit voraus. Die Qualitätsverbesserung wird durch Anpassung der Eigenschaften erreicht. Die Vorgaben sind in diesem Fall den Lebenszeitstatistiken gegenübergestellte Eigenschaften der Mikrosysteme. So kann für eine Auswahl von vorgegebenen Ausfallcharakteristiken eine Verfeinerung der Eigenschaften des Mikrosystemes erreicht werden.

Diese Ausührungen zeigen, dass ein breites Einsatzspektrum für Neuronale Netze in de Entwicklung von MEMS möglich ist. Der Einsatz Neuronaler Netze in der Entwicklung von MEMS verdeutlicht, dass ein breites Einsatzspektrum möglich ist. Dies ist der Vielzahl und der Anpassbarkeit von Netzwerkmodellen geschuldet. Der Trend geht in Richtung Mustererkennung und Vergleichen von Soll– und Ist–

verhalten von Strukturen. In dieser Arbeit wird die Mustererkennung eine tragende Rolle spielen.

Ein weiteres Verfahren, welches für die Entwicklung von Mikrosystemen eingesetzt werden kann, sei hier noch am Rande erwähnt. Hubbard und Antonsson [HA97] beschreiben den Einsatz Zellulärer Automaten zur Vorhersage anisotrop geätzter dreidimensionaler Strukturen in Silizium für beliebige Maskenformen. Zelluläre Automaten bestehen aus zahlreichen kleinen Zellen mit einem jeweils einfach zu beschreibenden Verhalten. Das gesamte Verhalten vieler Zellen spiegelt das komplexe Verhalten eines physikalischen Systemes wider. Hubbard unterteilt das Material Silizium in Zellen mit jeweils mehreren Siliziumatomen. Das Verhalten dieser und ihre Wechselwirkungen untereinander sind bekannt. Somit kann das Gesamtverhalten bei einem entsprechenden Ätzprozess simuliert werden.

Zelluläre Automaten finden bei Aufgaben Anwendung, die sich in viele gleiche einfache Teile zerlegen lassen und deren Wechselwirkung bekannt ist. Sie kommen in dieser Arbeit nicht zum Einsatz, da sich gezeigt hat, dass keine Aufspaltung in einfache gleiche Teilaufgaben möglich ist.

Da es im Entwicklungsprozess von MEMS zahlreiche Optimierungsaufgaben gibt, müssen die Lösungsstrategien entsprechend ausgewählt werden.

Kapitel 2

Grundlagen

Dieses Kapitel wird einen Einblick in die Grundlagen dieser Arbeit geben. Der Weg geht dazu von der Simulation und Fertigung eines MEMS über die Messung, beziehungsweise Ermittlung von Fertigungsparametern, zu Optimierungsansätzen für die Entwicklung.

Als erstes wird auf die Fertigungstechnologie Bonding and Deep Reactive Ion Etching (BDRIE) für MEMS eingegangen. Im darauffolgenden Abschnitt wird ihre Modellierung in ANSYS erklärt. Dabei werden charakteristische Parameter ungedämpfter Feder-Masse-Systeme erläutert, welche experimentell nach dem im anschließenden Abschnitt beschriebenen Verfahren gemessen werden.

Abschnitt 2.5 beschäftigt sich mit den Grundsätzen der Parameteridentifikation, welche die Regressionsanalyse aus dem vorhergehenden Abschnitt verwendet.

Die Grundlagen der Informationsverarbeitung und Optimierung werden in Abschnitt 2.6 umrissen, die darauffolgenden führen die Grundlagen für die in dieser Arbeit verwendeten Verfahren weiter aus.

2.1 Fertigungstechnologie - BDRIE

Für die Herstellung von MEMS in Siliziummikromechanik wird das BDRIE-Verfahren genutzt. Es ist eine oberflächennahe Siliziumbulktechnologie, mit welcher hohe Aspektverhältnisse zwischen 20 und 30 möglich sind. Das Verfahren kombiniert das Bonden eines vorstrukturierten Basiswafers und eines aktiven Wafers mit der Anwendung von tiefem reaktivem Ionenätzen (Deep Reactive Ion Etching (DRIE)) des aktiven Wafers. Im Gegensatz zu anderen Fertigungstechnologien, welche isotrope Unterätzung nutzen, um bewegliche Strukturen freizulegen, besitzt BDRIE weniger Designeinschränkungen. Die beweglichen Strukturen benötigen keine Löcher für die Unterätzung und die Breite der vertikalen Spalte kann zwischen $1,5\,\mu m$ und mehreren hundert μm variiert werden. [HKB+05]

Die Technologieparameter Ätzrate, Ätzselektivität und Anisotropie sind in den

Gleichungen 2.1, 2.2 und 2.3 definiert. [GD06]

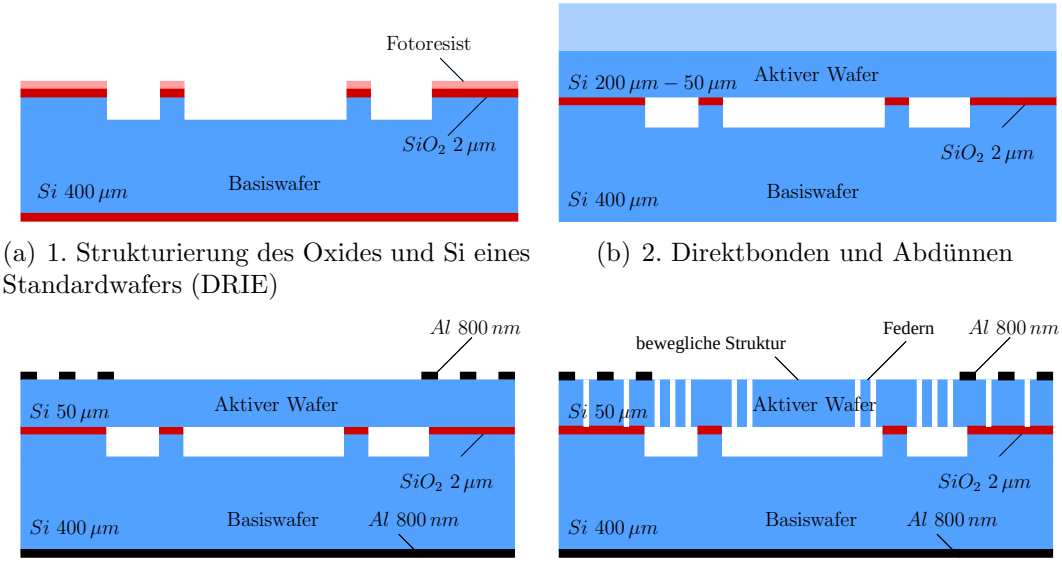
$$\text{Ätzrate } R = \frac{\Delta d}{t} \tag{2.1}$$

$$\text{Ätzselektivität } S = \frac{R_{\text{Si}}}{R_{\text{Maske}}} \tag{2.2}$$

$$\text{Anisotropie } A = 1 - \frac{R_{\text{lateral}}}{R_{\text{vertikal}}} \tag{2.3}$$

mit Δd ... geätzte Dicke
t ... Ätzzeit
R_{Si} ... Ätzrate Silizium
R_{Maske} ... Ätzrate Lithographiemaske
R_{lateral} ... laterale Ätzrate im Medium
R_{vertikal} ... vertikale Ätzrate im Medium

Für den Basiswafer kommt Glas oder Silizium zum Einsatz, für den aktiven Wafer Silizium. Der aktive Wafer enthält bewegliche Strukturen und die Antriebs- und Detektionselektroden für laterale Bewegungen. In Abbildung 2.1 sind die Prozessschritte für die Kombination Silizium-Silizium dargestellt. Im ersten Schritt erfolgt



(a) 1. Strukturierung des Oxides und Si eines Standardwafers (DRIE)

(b) 2. Direktbonden und Abdünnen

(c) 3. Metallisierung und Strukturierung der Al-Schicht

(d) 4. Trenchätzen des Si bis zur Kammer

Abbildung 2.1: Ablauf BDRIE — Beispiel Si-Si (nach [HKB+05])

die Strukturierung des Basiswafers und der isolierenden thermischen Oxidschicht durch DRIE. Dadurch werden die späteren, 50 μm tiefen, Kammern für die beweglichen Strukturen gebildet. Im nächsten Schritt erfolgt das direkte Bonden eines 200 μm dicken Si-Wafers, welcher durch Ätz- und Polierprozesse auf die Höhe der beweglichen mikromechanischen Strukturen (50 μm) abgedünnt wird. Anschließend wird die Aluminiumschicht auf den aktiven Wafer aufgebracht und strukturiert. Das

Trenchätzen des aktiven Wafers bis zu den Kammern beziehungsweise zum thermischen Oxid stellt die beweglichen Siliziumelemente frei. Die laterale Isolation wird durch die entstehenden Luftspalte gewährleistet.

Die Prozesstoleranzen (Abbildung 2.2) der resultierenden Strukturen werden durch die einzelnen Schritte geprägt. Die Strukturdicke wird durch den Abdünnungsprozess und die Total Thickness Variation (TTV) des Wafers in Schritt 2 bestimmt. Die TTV liegt im Bereich von ± 2 bis $5\,\mu m$. Sie ist bei industriell gefertigten Wafern geringer.

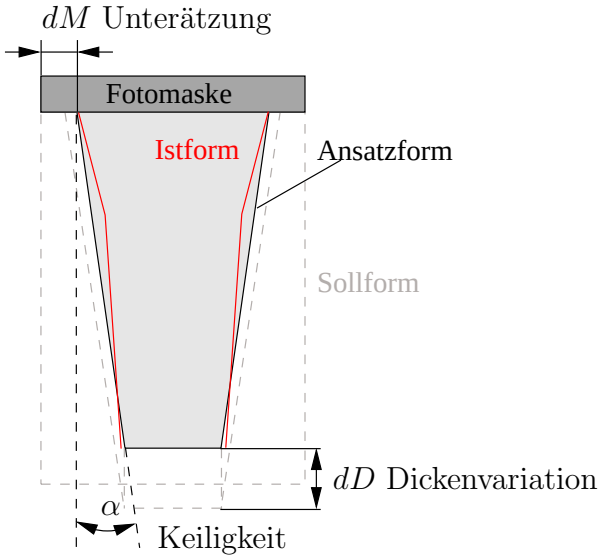


Abbildung 2.2: Prozesstoleranzen bei BDRIE

Durch den Bondprozess und das Aufbringen der Metallisierung kann mechanischer Stress in das Material eingebracht werden. Die Verformung des Chips kann dabei in einem sehr geringen Bereich bis zu $0,5\,\mu m$ liegen. Selbst kleine sind allerdings ausreichend, um einen Stress von ungefähr ± 5 bis $15\,MPa$ in MEMS zu verursachen. Führt dieser Stress zu einer Verschiebung der Eigenfrequenz von etwa 5 bis 20%, so kann das MEMS unbrauchbar werden. [Sha05]

Der DRIE-Prozess in Schritt 4 bestimmt die Querschnitte von Federn und Elektroden. Dieser Prozess soll ein hohes anisotropes Ätzverhalten, eine gute Homogenität und ausreichend Selektivität zur Ätzmaske haben. Die Time Multiplexed Deep Etching (TMDE)-Technik [ABL+99], welche sich aus wiederholenden Sequenzen von Ablagerungs- und Ätzschritten zusammensetzt, ist heute weit verbreitet. Während der Ablagerung wird C_4F_8 genutzt, um die Seitenwände zu passivieren, mit SF_6 wird das Ätzen umgesetzt. Eine aufführlichere Beschreibung findet sich in [Hil04].
Dieser Prozessschritt verursacht das typische Ätzprofil in Abbildung 2.2 mit Maskenunterätzung und der Keiligkeit des Querschnitts. Das Profil ist stark vom Prozesskammerdruck, der Fließgeschwindigkeit des Gases, der Zykluszeit, der angelegten Spannung und der Spaltbreite abhängig. [Hil04] Es ist weiterhin von der Temperatur in der Nähe der geätzten Oberfläche abhängig. Schmale MEMS-Elemente wie

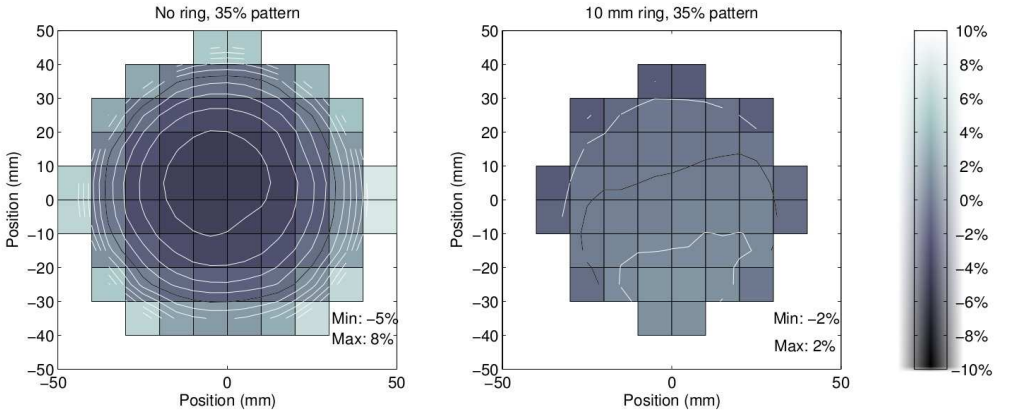
lange Federn führen die Wärme nicht von den ätzenden Stellen ab, daher ist die Abweichung in diesen Bereichen größer. [SSS+09]

Typische Werte für technologische Abweichungen sind für die an der Technischen Universität Chemnitz verwendeten Prozesse in Tabelle 2.1 dargestellt.

Tabelle 2.1: Werte für technologisch bedingte Abweichungen (BDRIE-Prozess)

Merkmal	Typische Abmessung	Typische Abweichungen vom Nennwert
Federbreite	$3 - 7\,\mu m$	$\pm 100 - 200\,nm$
Federhöhe	$50\,\mu m$	$\pm 1 - 3\,\mu m$
Keiligkeitswinkel	$0,1° - 0,3°$	$\pm 0,01° - 0,05°$
Mechanischer Stress	0	$\pm 5 - 15\,MPa$

Neben der Abweichung der Struktur kann es zu Schwankungen der Ätzrate über dem Wafer kommen. Die Ätzratenvariation steigt mit der Strukturdichte. Diese wird somit durch die Ätzratentoleranz limitiert. Wenn der Wafer sich während des Ätzprozesses in einer nichtätzbaren Umgebung befindet, kommt es zu einer ansteigenden Konzentration der ätzenden Flour-Radikale und damit zu einer Erhöhung der Ätzrate vom Wafermittelpunkt in Richtung der Waferperipherie. Dieser Effekt kann durch die Optimierung der Technologie [Lea06] oder über Kompensationsstrukturen [JJQH05] reduziert werden. Abbildung 2.3 stellt einen Wafer mit Ätzratenvariation und einem ohne gegenüber.



(a) Wafer ohne Kompensation (b) Wafer mit Kompensation (c) Ätzrate

Abbildung 2.3: Ätzratenverläufe über den Wafer

Trotz der Kompensationsmöglichkeiten muss die Veränderung der Prozessabweichung über den Wafer ermittelt werden, da diese Einfluss auf das Verhalten der MEMS haben. [CM07]

2.2 Modellierung in ANSYS

ANSYS ist ein Simulationsprogramm, das die Finite Elemente Methode (FEM) umsetzt. FEM ist ein numerisches Verfahren zur Lösung von Differentialgleichungen. Stark vereinfacht wird das Berechnungsgebiet in endlich viele kleine, aus Knoten bestehende, Elemente unterteilt. Auf diese Knoten werden Ansatzfunktionen aufgeprägt. Diese bilden mit Anfangs- und Randbedingungen ein Gleichungssystem. Seine Lösung entspricht der numerischen Lösung der Differentialgleichung. Die Methode wird für physikalische Aufgaben eingesetzt. Die Grundlagen und Anwendungen der Finiten Elemente Methode sind in [KW99] und [Kle00] ausführlich erläutert. Dieser Abschnitt soll einen kurzen Einblick in die Simulation mit ANSYS geben.

Die FE-Simulation in ANSYS erfolgt in drei Schritten:

- Modellerstellung und Aufbringen der Lasten (Präprozessor)
- Ausführen des Lösers (Lösungsprozessor)
- Anzeige und Auswertung der Ergebnisse (Postprozessor)

Die Modellerstellung erfolgt in dieser Arbeit über Solid-Modeling und nach dem Bottom-Up-Prinzip. Das bedeutet, dass die Vernetzung auf Basis der Modellgeometrie automatisch erfolgt. Die Geometrie wird ausgehend von Punkten über Linien, Flächen und Volumen erzeugt, also von unten nach oben erstellt.Weitere Modellierungsmöglichkeiten finden sich in [SGM06].

Für Strukturen der vorliegenden Arbeit wird vorrangig die Grundfläche, oder entsprechend der Symmetrie nur ein Teil davon, erstellt. Anschließend erfolgt eine Extrusion in Strukturtiefe und das Volumen wird mit Quaderelementen regelmäßig vernetzt. Die Netzteilung entlang von Linien steuert die Netzdichte. Bei symmetrischen Strukturen wird zusätzlich eine Spiegelung entlang der Symmetrielinien vorgenommen und Diskontinuitäten im Netz an der Schnittstelle behoben.

Für feste Einspannungen werden die Verschiebungen an den entsprechenden Knoten der Finiten Elemente auf Null festgesetzt. Anschließend erfolgt die Analyse — in diesem Fall Modalanalyse — im Lösungsprozessor und die Auswertung im Postprozessor.

Die in dieser Arbeit notwendige Parametervariation erfolgt in mehrere Schritten. Diese sind in Abschnitt 3.4 beschrieben. Die dabei berücksichtigte Keiligkeit von MEMS-Strukturen durch DRIE (siehe Abschnitt 2.1) ist in ihrer Umsetzung in Abschnitt 3.5 erläutert.

Mit der Berechnung der Eigenfrequenzen und Eigenformen (Modalanalyse) werden die wichtigsten Kenngrößen für lineare dynamische Untersuchungen ermittelt. Sie gehen aus der Betrachtung des freischwingenden Systems hervor. Gleichung 2.4 zeigt die Bewegungsgleichung eines ungedämpften Mehrmassenschwingers in Matrixschreibweise. N Freiheitsgrade führen zu einem homogenen Gleichungssystem mit N Gleichungen.

$$M\ddot{u} + Ku = 0 \tag{2.4}$$

mit M ... Massenmatrix
K ... Steifigkeitsmatrix
u ... Verschiebung / Weg
$\ddot{u}$ ... Beschleunigung

Bei Anregung mit einer harmonischen Funktion

$$u(t) = \overline{u} \cdot sin(\omega t)$$

ergibt sich bei konstanter Masse und Steifigkeit die für jeden Zeitpunkt gültige Gleichung 2.5. Die Eigenfrequenzen werden aus den Eigenwerten des Gleichungssystems berechnet. Die Schwingform $\overline{u}$ ergibt sich aus der Rückrechnung.

$$(K - \omega^2 M)\overline{u} = 0 \tag{2.5}$$

Die Amplituden der Eigenformen $\overline{u}$ sind unbestimmt, da jedes Vielfache Gleichung 2.5 erfüllt. Sie werden daher normiert — häufig auf den höchsten Wert der Auslenkung. In der Messung zeigt sich bei Eigenfrequenzen eine deutliche Überhöhung im Amplitudengang. Zur Kennzeichnung der Richtungen der maximalen Auslenkungen sind die Begriffe *In-Plane* und *Out-of-Plane* definiert. Bewegungen *In-Plane* finden fast ausschließlich in der Waferebene statt. Bei *Out-of-Plane* bewegt sich die Struktur aus ihr heraus. Dies ist für die Messbarkeit der Eigenmoden relevant.

Numerische Methoden zur Modalanalyse sind in ANSYS eingebettet, wie beispielsweise Block-Lanczos (`lanb`) und die Unterraum-Methode (subspace method subsp). Die Analyse ist linear, jegliche Nichtlinearität bleibt unberücksichtigt. Eine ausführlichere Beschreibung findet sich in [SGM06].

2.3 Messung von MEMS-Eigenschaften

Eigenmoden werden experimentell mit einem Laser Doppler Vibrometer (LDV) berührungslos bestimmt. Ein LDV arbeitet auf Basis von Interferometrie (Stichwort „Laserdopplerinterferrometrie“). Ein Laserstrahl wird auf eine schwingende Oberfläche fokusiert. Das zurückgestreute Licht weist eine Frequenzverschiebung durch den Doppler-Effekt auf. Diese Verschiebung wird ausgewertet und beispielsweise als Wegverschiebung dargestellt. [Pola] Eine Erweiterung stellt die Laser-Scanning-Vibrometrie dar, mit welcher sich schwingende Flächen erfassen lassen. Als Messsystem wird der in Abbildung 2.4 dargestellte MSA-500 Micro System Analyzer der Firma Polytec eingesetzt. Er ist für die Messung von MEMS optimiert und umfasst unter anderem ein auf Laser-Scanning-Vibrometrie basierendes Messsystem, sowie ein Auswertesystem. Out-of-Plane und In-Plane-Schwingungen sind messbar. Weitere Daten finden sich auf der Hersteller-Webseite [Polb].

Die Anregung der zu messenden Strukturen erfolgt in einem elektrostatischen Streufeld. Piezoshaker werden wegen der geringeren erreichbaren Anregungsfrequenz nicht eingesetzt. Abbildung 2.5 zeigt eine Struktur in der Messkammer.

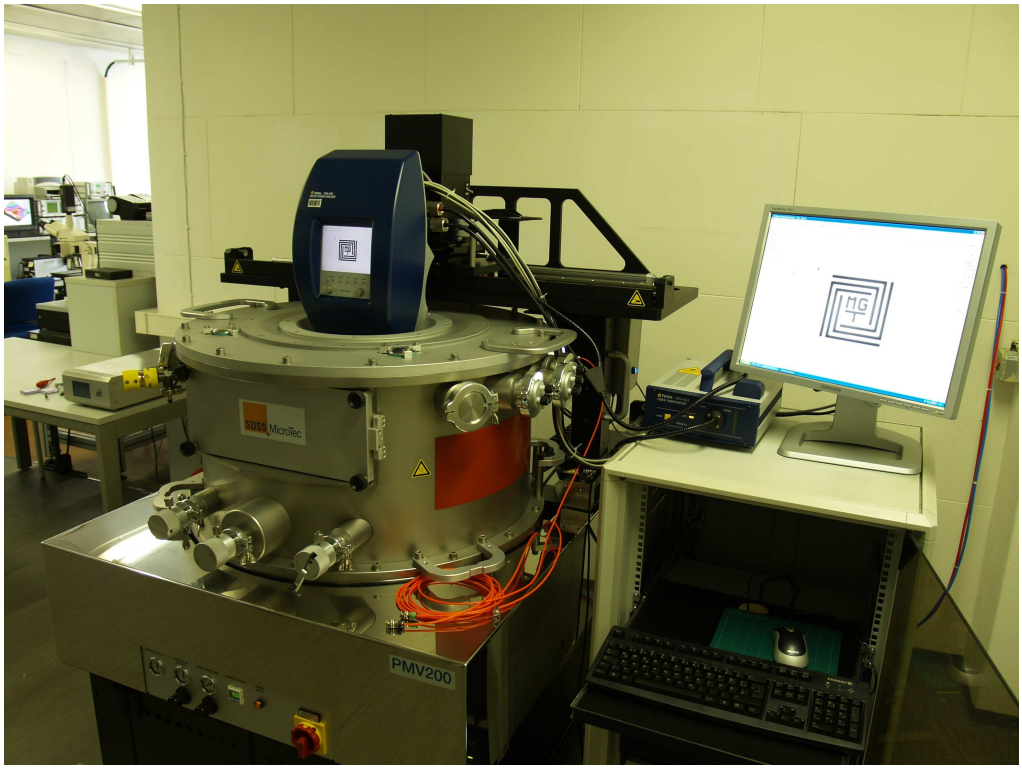


Abbildung 2.4: Polytec MSA-500 Micro System Analyser

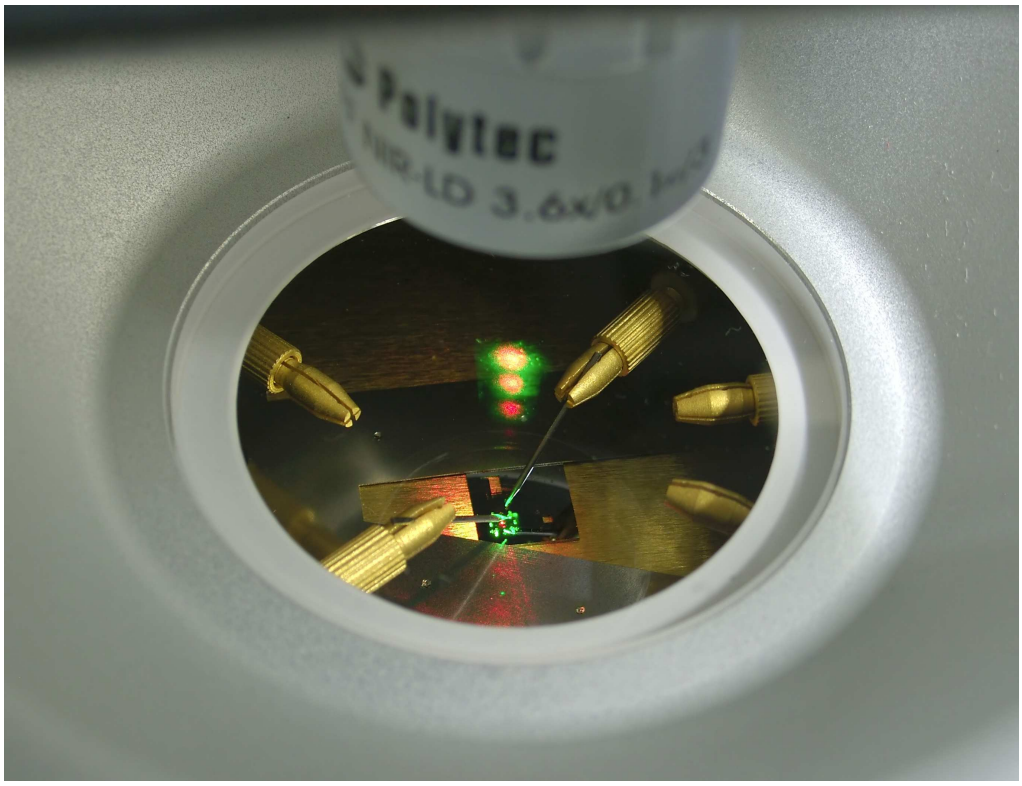


Abbildung 2.5: Strukturen in der Messkammer

Tabelle 2.2: Richtwerte der Messintervalle von Eigenmoden

Frequenzmessbereich f_{mess} / kHz	messbare Eigenmoden	Bemerkungen
$0 < f_{mess} \leq 10$	In-Plane, Out-of-Plane	Messung erfolgt stroboskopisch
$10 < f_{mess} \leq 100$	In-Plane, Out-of-Plane	Messung erfolgt durch LDV
$100 < f_{mess} \leq 500$	Out-of-Plane	In-Plane nicht mehr eindeutig messbar
$f_{mess} > 500$	-	Messungen sind nicht mehr zuverlässig möglich

Ein Verfahren zur Parametercharakterisierung von MEMS ist in [Mei07] beschrieben. Die messtechnische Seite steht in dieser Arbeit im Hintergrund. Die Beschränkung des Messverfahrens werden aber in der Evaluierung der Teststrukturen berücksichtigt.

Neben den Messparametern des Herstellers wurden in Experimenten Grenzen der Messbarkeit festgestellt. Diese sind in Tabelle 2.2 dargestellt. Die Werte sind allerdings Richtwerte, es ist durchaus möglich, Eigenmoden außerhalb der angegeben Intervalle zu messen. Generell wird deutlich, dass die Messbarkeit mit steigender Frequenz abnimmt.

Um höhere Moden (ab 250 kHz) zu erregen, wird ein Hochspannungsverstärker benötigt. Dieser und die Elektroden können eine elektrostatische Federerweichung, in Abhängigkeit von Elektrodenform und Abstand, verursachen. Dieser Effekt wurde bei Teststrukturen noch nicht untersucht, allerdings bei anderen Strukturen beobachtet (siehe [BHY+02] und [SMR+04]).

2.4 Regressionsanalyse

Die Regressionsanalyse ist ein statistisches Verfahren, mit dem es möglich ist, einen Zusammenhang zwischen einer abhängigen Variablen Y und einer oder mehreren unabhängigen Variablen x_i mit

$$Y = f(x_1, \ldots, x_n)$$

herzustellen.

Es existieren verschiedene Modelle, um die Abhängigkeiten zu charakterisieren. Ferner wird zwischen einfacher Regression (eine unabhängige Variable) und multipler Regression (mehrere unabhängige Variablen) unterschieden. An dieser Stelle soll nur letztere betrachtet werden, da sie den allgemeinen Fall darstellt.

Das lineare Modell ohne Wechselwirkungen setzt voneinander isolierte unabhängige Variablen voraus. Damit treten keine Kreuzterme in der Regressionsfunktion auf (siehe Gleichung 2.6). Treten Wechselwirkungen zwischen den un-

abhängigen Variablen auf, so werden die entsprechenden Kreuzterme hinzugefügt.

$$Y = f(x_1, \ldots, x_m) = b + \sum_{i=1}^{m} a_i \cdot x_i \tag{2.6}$$

mit Y ... abhängige Variable
x_i ... i-te unabhängige Variable
m ... Anzahl der unabhängigen Variablen
a_i ... Regressionskoeffizient der i-ten unabhängigen Variablen
b ... Gleichanteil
n ... Anzahl der unabhängigen Variablen

Quadratische und höherwertige Modelle mit Wechselwirkungen lassen sich mit Gleichung 2.7 beschreiben.

$$Y = f(x_1, ..., x_m) = \sum_{\substack{n_i \in \{0, ..., N_{x_i}\} \\ k \in \{0, ..., K\}}} \prod_{i=1}^{m} a_k \cdot p_i^{x_i} \tag{2.7}$$

mit N_{x_i} ... Regressionsgrad des Parameters x_i
a_k ... Koeffizient des Regressionsgliedes k
k ... Regressionsglied
K ... Regressionspolynomgrad
$K = \prod_{i=1}^{m} (1 + N_{x_i})$

Die Regressionskoeffizienten a werden über die Methode der kleinsten Fehlerquadrate ermittelt. Das Verfahren ist in Anhang A ausführlicher für die konkrete Anwendung erläutert.

Das Ergebnis der Regressionsanalyse ist das sogenannte Response Surface (auch Antwortfläche), welches einen funktionalen Zusammenhang zwischen der abhängigen und den unabhängigen Variablen herstellt.

2.5 Identifikation von Parametern

Die Parameteridentifikation von MEMS basiert, wie in Kapitel 1 erwähnt, auf drei Schritten:

1. Numerische Modalanalyse der (mit mechanischem Stress vorbelasteten) TS
2. Experimentelle Modalanalyse der TS
3. Abgleich von numerischer und experimenteller Modalanalyse und Extraktion der Parameter

Die Grundlage ist nicht das eigentliche Mikrosystem, sondern die für diesen Zweck optimierten Teststrukturen.[SFS+08]

Wie in Abbildung 2.6 dargestellt, sind die Schritte zur Ermittlung der theoretischen und realen Antwort einer Teststruktur unabhängig voneinander. Die Numerische Modalanalyse liefert über eine FE-Simulation der Teststruktur mit verschiedenen Werten typischer Designparameter Werte für das Erstellen einer mehrdimensionalen Regressionsfunktion (Response Surface). Die konkrete Vorgehensweise ist in Abschnitt 3.4 beschrieben. Die experimentelle Modalanalyse erfolgt durch Messung der gefertigten Teststrukturen mit einem LDV und einem Signalanalysator wie in Abschnitt 2.3 beschrieben. Die ermittelten Eigenfrequenzen und die zugehörigen Bewegungsformen bilden die reale Antwort der Teststruktur.

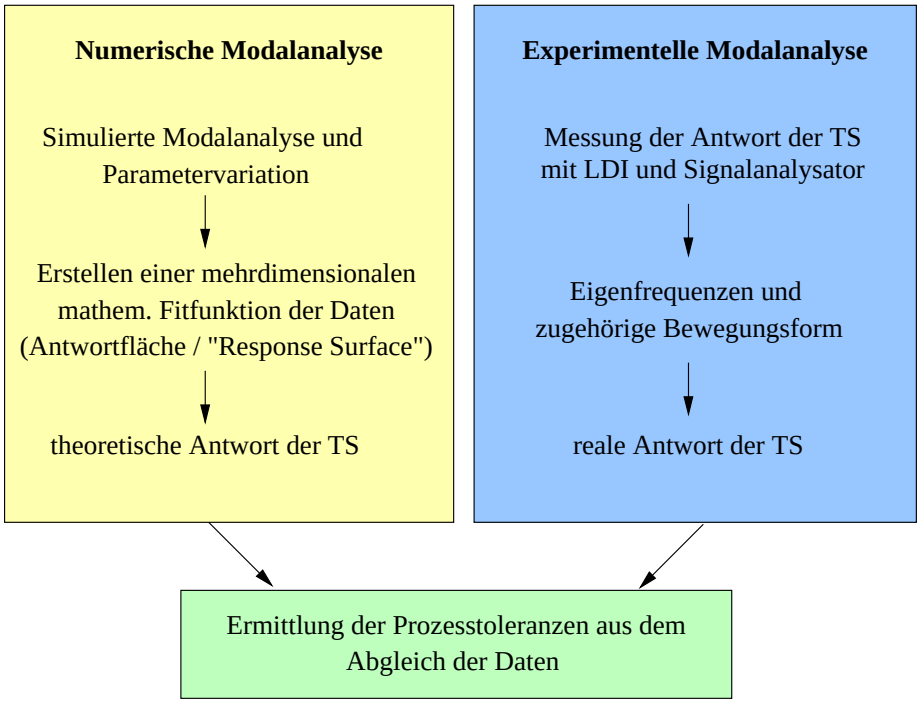


Abbildung 2.6: Schritte der Parameteridentifikation

Durch Kombination der theoretischen und realen Werte werden die unbekannten gesuchten Parameter durch die Methode der kleinsten Fehlerquadrate ermittelt. Der Fehler für die gesuchten Parameter p muss nach Gleichung 2.8 minimal werden.[SFS+08]

$$Res(\vec{p}) = \sum_{i=1}^{N} \left(\frac{f_{\mathrm{FEM}i}(\vec{p}) - f_{\mathrm{exp}i}}{f_{\mathrm{exp}i}} \right)^2 \qquad (2.8)$$
$$Res(\vec{p}) \rightarrow min$$

mit $\vec{p}$ ... Vektor der gesuchten Parameter
N ... Anzahl der realen Eigenfrequenzen
$f_{\mathrm{FEM}i}$ ... Soll-Eigenfrequenz aus der FEM-Simulation
$f_{\mathrm{exp}i}$ ... reale Eigenfrequenzen aus der Messung

Die Anzahl der benötigten Eigenfrequenzen ist abhängig von der Anzahl der zu erfassenden Designparameter. Diese Eigenfrequenzen müssen unterschiedliche

Eigenmoden besitzen.

Bei der Variation von Parametern kann es zum Vertauschen von Eigenmoden kommen (siehe Anhang C). Die Eigenmoden müssen für die Antwortfläche eine konstante Reihenfolgen aufweisen, da sonst der Regressionsfehler steigt. Es ist somit eine Sortierung der Eigenmoden unter Umständen notwendig.

2.6 Optimierung und Informationsverarbeitung

Die Optimierungstheorie ist ein Teil der angewandten Mathematik, dass sich mit der Optimierung bestimmter Prozesse und Verfahren beschäftigt. Teilgebiete sind beispielsweise die lineare und nichtlineare Optimierung.

Die Lösungsverfahren umfassen klassische Optimierungsverfahren ebenso wie heuristische Verfahren. Diese haben das Ziel, mit begrenztem Wissen und Zeit zu einer guten Lösung zu kommen. Die Lösungssuche erfolgt in einem n-dimensionalen Parameterraum mit den Parametern $p_1, \ldots, p_n$ und einer Qualitätsfunktion — auch Qualitätsgebirge — Q, welche jede Parameter-Kombination eindeutig bewerten kann. Gesucht sind die optimalen Parameterwerte $p_{i,\text{optimal}}$, für die $Q(p_1, \ldots, p_n)$ maximal wird. Abbildung 2.7 zeigt das Qualitätsgebirge eines 2-dimensionalen Parameterraums.

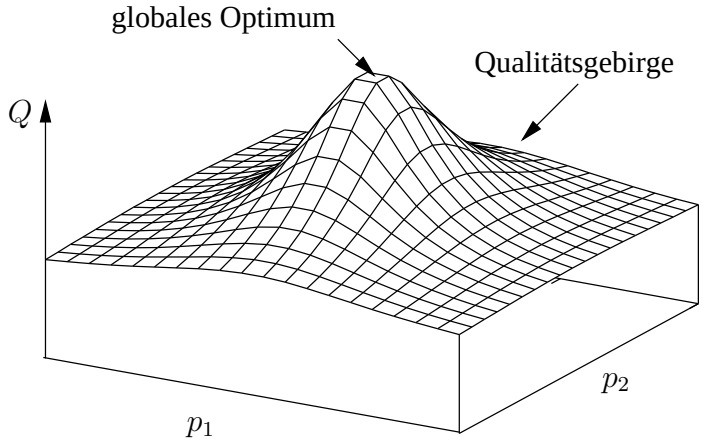


Abbildung 2.7: Qualitätsgebirge eines 2D-Parameterraums

Klassische Optimierungsverfahren lassen sich in *deterministische* und *nichtdeterministische Verfahren* untergliedern. Evolutionäre Algorithmen (EA) zählen zu den *heuristischen Verfahren*, welche allerdings nicht direkt in diese Einteilung einordnen lassen. Im folgenden sollen alle drei kurz vergestellt und auf die Eignung in dieser Arbeit hin verglichen werden.[SHF94]

Deterministische Verfahren, welche sich auch für die Suche in hochdimensionalen Parameterräumen eignen, sind unter anderem die Gauß-Seidel Strategie, das Gradienten-Verfahren und das Simplex-Verfahren. Sie können unter dem Stichwort "hill-climbing Stategien" zusammengefasst werden, da sie sich wie Bergsteiger, im-

mer am lokalen Anstieg orientieren und bevorzugt dorthin bewegen, wo ein Anstieg festzustellen ist.[SHF94]

Die Gauß-Seidel Strategie geht sukzessiv vor. Der Parameter p_1 wird in eine Richtung verändert. Steigt die Qualitätsfunktion, wird der Parameter weiter in diese Richtung verändert, bis sein maximal zulässige Wert erreicht ist oder Q wieder "kippt", also kleiner wird. Die Prozedur wird nacheinander für alle Parameter durchgeführt und solange wiederholt, bis ein zufriedenstellender Wert für Q erreicht ist.[SHF94]

Das Gradienten-Verfahren richtet sich in jedem Punkt des Qualitätsgebirges nach dem steilsten Tangentenanstieg. Dazu werden die partiellen Ableitungen der Qualitätsfunktion nach den Parametern bestimmt. Die Parameteranpassung erfolgt in Richtung des steilsten Gradienten und proportional zur Steigung, das heißt in der Nähe des Optimums erfolgen kleine Adaptionsschritte, in weiterer Entfernung in der Regel größere. In der Praxis ist es nicht auf differentierbare Funktionen beschränkt, da oft mit Differenzenquotienten statt mit Ableitungen gearbeitet werden kann. [SHF94]

Das Simpex-Verfahren geht nicht von einem Startpunkt aus, sondern von mehreren gleichzeitig. In einem n-dimensionalen Suchraum werden $n+1$ Startpunkte verwendet und so gesetzt, dass sie untereinander gleiche Abstände aufweisen. Der Eckpunkt des Simplex mit dem schlechtesten Wert von Q wird gestrichen, an seine Stelle tritt ein neuer Punkt, der durch Spiegelung des gestrichenen Punktes am Mittelpunkt des verbleibenden n-Ecks hervorgeht. Es kann zu einer Oszillation kommen, wenn der neue Punkt wieder der schlechteste ist. Um dies zu verhindern, fällt die Wahl zur Spiegelung in diesem Fall auf den zweischlechtesten Punkt. Nach einigen Iterationen werden noch Polyeder erzeugt, die um den Eckpunkt mit dem höchsten Qualitätswert Q rotieren. Eine Qualitätsverbesserung kann dann nur noch erreicht werden, indem die Kantenlänge des Polyeders reduziert wird.[SHF94]

Nichtdeterministische Optimierungsverfahren machen bei der Suche nach dem Optimum in großen Suchräumen vom Zufall Gebrauch. Wenn über Lage des Optimums keine Kenntnis vorhanden ist — was in der Praxis häufig der Fall ist, sind zufallsgesteuerte Verfahren unter Umständen weitaus effizienter als deterministische Algorithmen. Basiert das deterministische Verfahren auf einer auch nur geringfügig falschen Annahme, kann es bereits nutzlos sein. Bei fehlender Lagekenntnis des Optimums ist oft nicht beurteilbar, ob Suchstrategie und Annahmen richtig sind. Ein Beispiel für ein nichtdeterministisches Verfahren ist das Monte-Carlo-Verfahren.[SHF94]

Heuristische Optimierungsverfahren liegen häufig in der Mitte der vorangegangenen Klassifizierung. Ein einfaches Beispiel ist das in Abschnitt 2.7 erklärte Trial-and-Error Verfahren. Bei diesem werden solange Lösungsmöglichkeiten getestet, bis ein akzeptables Ergebnis vorliegt. Diese ist oft nicht die optimale Lösung.

Die Grundidee der Evolutionäre Algorithmen basiert auf der natürlichen Evolution der Arten. Evolution ist die Veränderung der vererbbaren Merkmale von Lebewesen von Generation zu Generation. In der Natur streben die Lebenwesen nach einer Anpassung an die gegebenen Lebensbedingungen um verfügbare Ressourcen

besser nutzen zu können. Die zu optimierenden Parameter bilden in der Optimierungstheorie die Erbinformation beziehungsweise die Chromosomen. Durch Selektion werden bevorzugt bewährte, erfolgreiche Genome miteinander kombiniert. Evolutionäre Algorithmen suchen nicht mit konstanter Wahrscheinlichkeit den Suchraum ab, sondern bevorzugt in Regionen in denen eine überdurchschnittliche Zunahme erfolgreicher Genome zu erwarten ist. Sie sind also zielgerichteter als klassische Optimierungsverfahren, aber zählen dennoch nicht zu den deterministischen Verfahren. Diese Mischung ist das Erfolgsrezept.[SHF94] Ein weiterer Vorteil Evolutionärer Algorithmen im Vergleich zu "hill climbing" Methoden ist, das keine Differential- oder Differenzenquotienten benötigt werden.

In der Simulation wie auch in der experimentellen Analyse müssen Informationen verarbeitet werden. Soll eine Mustererkennung erfolgen, so geschieht dies im konventionellen Sinne durch den Mensch. Dies kostet Zeit und Ressourcen, deshalb ist ein Verfahren wünschenswert, welches Muster erkennt und entsprechend behandelt. In dieser Arbeit werden Neuronale Netze dafür eingesetzt. Die entsprechenden Grundlagen werden in Abschnitt 2.9 behandelt. Sie werden im Zusammenhang mit Optimierungsverfahren aufgeführt, da sie ein gegebenes Muster mit einem Zielmuster vergleichen können. Damit wäre im weitesten Sinne ein automatisiertes Trial-and-Error Verfahren möglich.

In der vorliegenden Arbeit werden Evolutionäre Algorithmen zur Optimierung von Teststrukturgeometrien eingesetzt, die Grundlagen finden sich in Abschnitt 2.8. Die Wahl fiel auf dieses Verfahren, da der Suchraum nicht durch eine Funktion definiert werden kann und Trial-and-Error nicht methodisch gut genug durchführbar ist. Neuronale Netze dienen zur Mustererkennung für den Simulationsteil der Parameteridentifikation, um die Reihenfolge der Eigenmoden zu prüfen beziehungsweise zu korrigieren (siehe Abschnitt 2.5).

2.7 Trial and Error - Versuch und Irrtum

Das Trial and Error-Verfahren ist eine Lösungssuche, bei der solange Lösungsmöglichkeiten getestet werden, bis die gewünschte Lösung gefunden ist. Dabei werden Fehlschläge in Kauf genommen. Theoretisch können damit alle Lösungen in endlich langer Zeit gefunden werden. Der Nachteil dieser Methode ist der zeitliche Aufwand bei größeren Lösungsräumen. Dieser kann allerdings mit entsprechender Erfahrung und Wissen eingeschränkt werden.

In Abbildung 2.8 ist beispielhaft ein Teil eines Suchbaumes dargestellt. Nach dem Verknüpfen von Einzelelementen zu komplexeren Elementen werden diese getestet und weiter modifiziert. Einige davon sind nicht erfolgversprechend (gekennzeichnet mit `Error`). Werden deren Unterbäume bei der Lösungssuche nicht berücksichtigt, kann der Lösungsraum enger eingegrenzt werden. Im Ergebnis dieses Verfahrens finden sich Strukturen, die den gestellten Randbedingungen genügen. Aus diesen kann eine weitere Auswahl erfolgen.

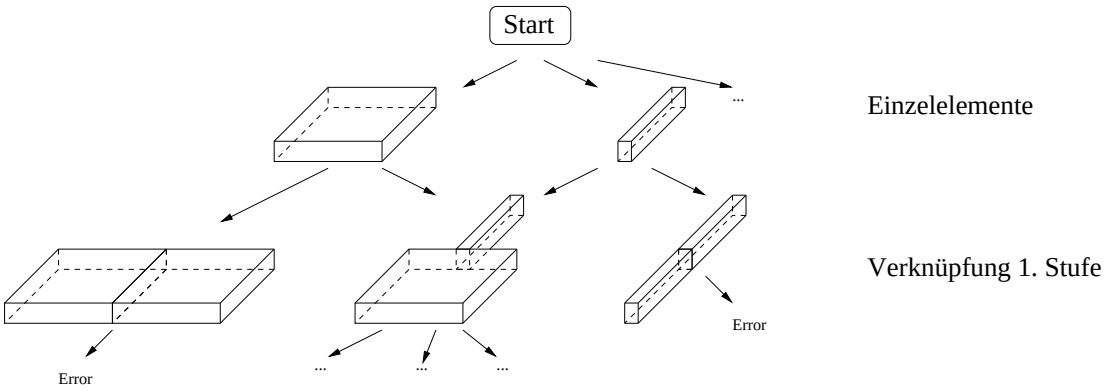


Abbildung 2.8: Beispieldarstellung Trial and Error-Verfahren

2.8 Evolutionäre Algorithmen

Die Natur ist Vorbild für eine gute Anpassungsstrategie. Milliarden von Lebensformen sind an ihre Umwelt im Laufe der Zeit durch Evolution angepasst worden. Daraus kann ein leistungsstarkes Optimierungsverfahren entwickelt werden, jedoch ohne exakt nachzubilden, sondern indem Grundprinzipien wie Selektion, Rekombination, Mutation etc. zur Lösung schwieriger theoretischer und praktischer Probleme genutzt werden.

In den folgenden Unterabschnitten wird die Grundlage Natur kurz umrissen und danach der Schwerpunkt auf die Evolution als Optimierungsverfahren gelegt. Es folgen Grundbegriffe und ihre Zusammenhänge untereinander. Anschließend werden die klassischen Modelle Evolutionärer Algorithmen zusammengefasst. Der letzte Unterabschnitt beschäftigt sich mit der programmtechnischen Umsetzung in die Bibliothek GAlib, welche in dieser Arbeit verwendet wird.

2.8.1 Grundlage Natur

Charles Darwins (1809-1882) Veröffentlichung „Über die Entstehung der Arten durch natürliche Auslese“ war 1859 eine Sensation. Er behauptete, dass sich alle Lebewesen über lange Zeiträume aus primitiveren Arten heraus entwickelt haben. Dabei wächst ihre Anzahl ohne hindernden Einfluss in der Regel in einem geometrischen Verhältnis an, während die Nahrungsmittelgrundlage meist nur arithmetisch, also wesentlich langsamer zunimmt. Daraus entsteht ein Selektionsdruck, der die Abnahme der Anzahl erzwingt, bis die Nahrungsmittel wieder ausreichen. Die am besten angepassten Individuen überleben (engl. *survial of the fittest*). Die Natur strebt nach einem Gleichgewicht der Arten.[SHF94]

2.8.2 Evolution als Optimierungsprozess

Die Natur ist verschwenderisch mit Individuen, wenn es daraum geht, eine Anpassung an die Lebensbedingungen zu erreichen. Die entscheidende Größe in die-

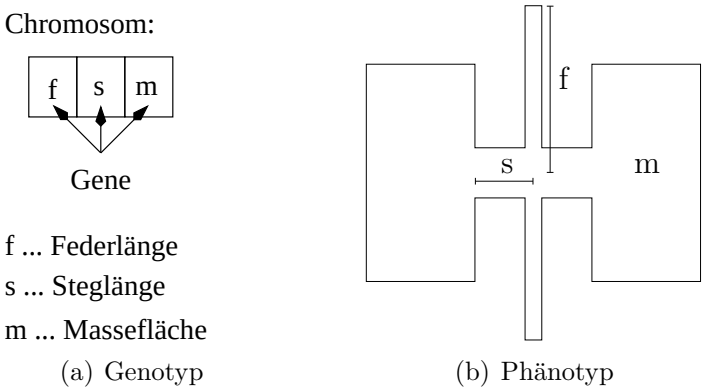
sem Prozess ist die Zeit. Für eine effiziente Optimierungsstrategie gibt es nur zwei Möglichkeiten — sehr kurze Generationsfolgen, damit sich die Individuen in einer Folge schnell den veränderten Bedingung anpassen können, oder viele Individuen gleichzeitig, damit die benötigte Evolutionszeit reduziert wird. Im Sinne der klassischen Suchstrategien ist dies eine gekoppelte Tiefen- und Breitensuche beziehungsweise eine Kombination aus serieller und paralleler Suche. Das ist für die Parallelisierbarkeit der evolutionären Suche entscheidend. Die Simulation kann dadurch erheblich beschleunigt werden.

Zusammenfassend stellt Evolution ein spezielles Optimierungsverfahren dar, dass in der Lage ist, selbst komplexeste Strukturen in relativ kurzen Zeiträumen an ihre Umwelt- und Lebensbedingungen anzupassen. Dies wird durch Manipulation der Erbinformation und den damit verbundenen grundlegenden Steuerungsmechanismen erreicht. Es ist eine Art Suchprozess im Raum der genetischen Informationen beziehungsweise im Raum der möglichen Erbanlagen. Ziel ist es, die Erbanlagen zu finden, die ein Individuum oder eine Art am besten dazu befähigen, mit den gegebenen Bedingungen zurecht zu kommen.

2.8.3 Grundbegriffe

Die Grundbegriffe für EA sind in der VDI/VDE Richtlinie 3550 Blatt 3 [Nor03] zusammengefasst. An dieser Stelle sollen einige erklärt werden, die für das Verständnis notwendig sind.

Ein EA besteht aus einer *Population*. Das ist eine Gruppe von Individuen mit gleichartiger Genomstruktur. Die *Populationsgröße* entspricht der Anzahl der Individuen. Sie sind Träger der genetischen Informationen (*Chromosomen*), wie in Abbildung 2.9 dargestellt.



(a) Genotyp (b) Phänotyp

Abbildung 2.9: Kodierung der Chromosomen

Ein *Genom* oder *Chromosom* ist aus *Genen* aufgebaut. Gene stellen in der Regel jeweils einen Parameter dar. Diese Repräsentation wird als *Genotyp* bezeichnet. Die konkrete Ausprägung der duch den Genotyp kodierten Eigenschaften ist der *Phänotyp*. Die Gesamtheit aller Gene in einer Population bildet ihren *Genpool*.

Mögliche Ausprägungen eines Gens, wie beispielsweise bei binärere Repräsentation 0 und 1, heißen *Allele*.

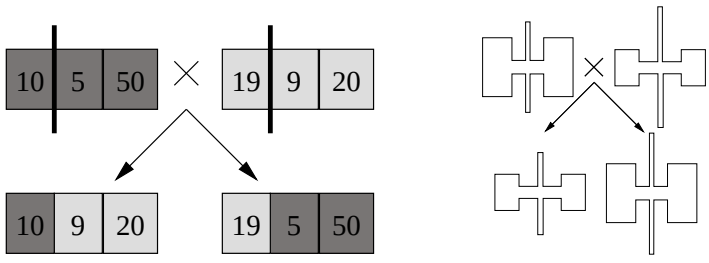
Die Anpassung an ein Zielkriterium erfolgt iterativ in *Generationen*. Der Evolutionsprozess wird durch das *Abbruchkriterium* beendet. Dies kann eine maximale Anzahl an Generationen, eine maximale Rechenzeit oder *Konvergenz* sein. Die Konvergenz wird in Gen-Konvergenz und Konvergenz gegen Optimalzustände unterschieden.

Der Suchraum wird durch eine *Fitnesslandschaft* (auch *Gütegebirge*) charakterisiert. Sie beschreibt den Zusammenhang zwischen der zu optimierenden Funktion (*Zielfunktion*) und der Lage des Individuums im Suchraum. Der Zielfunktionswert ist die *Bewertung* des Phänotyps eines Individuums.

Der Ablauf in einer Generation gliedert sich prinzipiell in die fünf Schritte *Selektion*, *Bildung*, *Manipulation*, *Bewertung* und *Ersetzen* von Individuen. Die erste Generation wird durch eine Urpopulation gebildet.

Die *Selektion* erfolgt auf Basis der *Fitness*. Diese gibt an, wie gut ein Individuum zur Reproduktion geeignet ist. Generell ist die Fitness von der Bewertung zu unterscheiden. Es gibt allerdings Varianten von EA, die diese Unterscheidung nicht treffen. Der *Selektionsdruck* ist über das Verhältnis der Wahrscheinlichkeit der Auswahl des besten Individuums zur durchschnittlichen Selektionswahrscheinlichkeit aller auswählbaren Individuen definiert. Die Selektion gibt die Richtung der Evolution an.

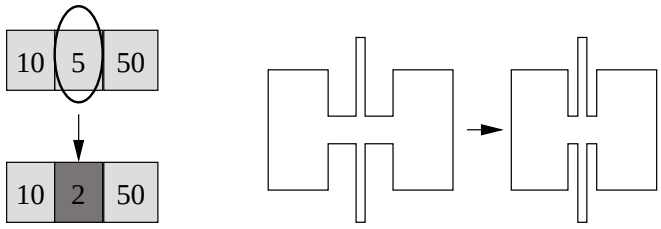
Die Bildung von Individuen kann durch *Replikation* oder *Rekombination* erfolgen. Bei ersterer werden sie kopiert, letztere erzeugt in der Regel aus zwei *Eltern* durch *Kreuzung* (auch *Crossover*) zwei *Nachkommen*. Abbildung 2.10 zeigt als Beispiel eine 1-Punkt-Kreuzung. Die Chromosomen der Eltern werden jeweils an einem Punkt getrennt und die Teile vertauscht wieder zusammengesetzt. Die Wahrscheinlichkeit, dass eine Kreuzung stattfindet, wird mit *Kreuzungs-* oder *Crossoverrate* bezeichnet.



(a) Genotyp (b) Phänotyp

Abbildung 2.10: Ein-Punkt-Crossover

Die Manipulation von Individuen bringt eine Variation in die Population. Sie erfolgt durch *Mutation*, bei der ein oder mehrere Gene eines Chromosoms verändert werden. Abbildung 2.11 zeigt beispielhaft eine Punktmutation. Die *Mutationsrate* gibt die Wahrscheinlichkeit für das Auftreten einer Mutation an.



(a) Genotyp (b) Phänotyp

Abbildung 2.11: Ein-Punkt-Mutation

Die Bewertung legt die Lage der Individuen im Gütegebirge fest. Auf Grund dieser werden die Individuen für die folgende Generation bestimmt. Die *Bewertungsfunktion* ist eine Art Güte. Sie gibt an, wie nahe ein Chromosom am gesuchten optimalen Wert ist.

Werden die Details vernachlässigt, beruht der von der Evolution durchgeführte Suchprozess auf drei einfachen Prinzipien: der Mutation des Erbgutes, der Rekombination der Erbinformation und der Selektion auf Grund der Tauglichkeit eines Individuums. Die Evolution kombiniert einen ungerichteten Suchprozess mit einem gerichteten. Die Mutation ist ungerichtet, ihr Sinn liegt in der Erzeugung von Varianten und Alternativen. Aus Sicht der Optimierungstheorie kommt ihr die Aufgabe zu, lokale Optima zu überwinden. Mutationen steuern den Suchprozess nicht aktiv, sie verhindern nur das Abdriften in einen suboptimalen Raum. Die Selektion ist für die eigentliche Steuerung der Evolution verantwortlich. Sie bestimmt die Richtung, in die sich das Erbgut verändert. Dabei legt sie fest, welche Phänotypen sich stärker vermehren und welche weniger stark. Dadurch wird die grundlegende Ausprägung und Ausrichtung eines Genoms einer Art festgelegt. Die Selektion wäre ohne Störungen (Mutationen) eine deterministische Komponente innerhalb der Evolution. Die Rekombination liegt hinsichtlich des Beitrags zur Zielfindung zwischen Mutation und Selektion und folgt festgelegten statistischen Regeln.

Aufbauend auf diesen Grundbegriffen folgt im nächsten Abschnitt die Darstellung klassischer Modelle Evolutionärer Algorithmen.

2.8.4 Klassische Modelle

Es gibt im Wesentlichen zwei Modelle Evolutionärer Algorithmen in der Computersimulation — die *Evolutionsstrategien* (ES) und die *Genetischen Algorithmen* (GA). Beide wurden in den 1960er Jahren unabhängig voneinander entwickelt.[SHF94]

Ingo Rechenberg entwickelte die *Evolutionsstrategien* an der Technischen Universität Berlin. Er hält eine naturgetreue Kopie der Evolution nicht für sinnvoll und sieht sie nur als Richtschnur für die Entwicklung eines leistungsstarken Such- und Optimierungsverfahrens. Rechenberg erstellte eine eigene Notation auf der Grundlage einer kompakten grafischen Darstellungsform. Die Kodierung eines Genoms erfolgt durch einen Vektor reeller Zahlen, auf die Genomstruktur an sich wird nicht weiter eingegangen. Populationen sind lediglich eine Menge solcher Vektoren. Es

gibt nur zwei Werte, die einen Informationswert enthalten — die konkreten Werte der zu optimierenden Parameter und die Werte der Qualitätsfunktion Q. Dies ist ein phänotyporientierter Ansatz. Im Allgemeinen werden aus einem Elter Nachkommen dupliziert. Danach werden diese durch Mutation modifiziert und anschließend der Ausgangsvektor (Elter) und die mutierten Duplikate bewertet — der Beste überlebt.[SHF94]

Evolutionsstrategien lassen sich somit auf die drei Schritte Selbstreproduktion, Mutation und Selektion zusammenfassen. Sie werden wiederholt, bis die Qualität eines Nachkommens ausreichend gut ist. Dieser Prozess ist seriell, es gibt zahlreiche Varianten der Evolutionsstrategien, auch solche, die eine Parallelisierung ermöglichen. Der Mutation kommt eine hervortretende Rolle zu, spezielle Rekombinations- und Selektionsstrategien stehen im Hintergrund. Sie erfolgt nach der statistischen Normalverteilung. Die Selektion von Individuen, Rekombination und Mischung von Populationen folgen dem Prinzip der statistischen Gleichverteilung. Damit ist die Selektion zur Nachkommenerzeugung unabhängig von der individuellen Fitness beziehungsweise wird nicht zwischen Fitness und Bewertung unterschieden.[SHF94]

Die *Genetischen Algorithmen* wurden von John Holland am Massachusetts Institute of Technology (MIT) entwickelt. Für sie gibt es keine formale Notation. Das Grundgerüst ist gleichartig mit dem der ES, die Unterschiede liegen im Detail. Es gibt ebenfalls viele Varianten.[SHF94]

Auf die Kodierung der Chromosomen wird gesteigerten Wert gelegt. Sie bestand in den Anfängen aus binären Vektoren, später kamen reellwertige Vektoren hinzu.[DeJ06]

Die Bewertungsfunktion legt die Optimierungskriterien und -ziele fest. Sie ist vergleichbar mit der Qualitätsfunktion der ES und immer problemspezifisch. Die Bewertung wird in eine Fitness umgerecht, anhand derer entschieden wird, mit welcher Wahrscheinlichkeit ein Individuum am Prozess der Nachkommenerzeugung teilnimmt. Direkte Transformationen sind ebenso möglich wie lineare und andere.[SHF94]

Ein weiterer Unterschied zu ES sind die angepassten Rekombinationsschemata. In ihnen kann problemspezifisches prozedurales Wissen über den Suchraum abgelegt werden. Mutationen sollen eine frühzeitige Konvergenz verhindern und damit dem Selektion und den Bewertungen, die im Laufe von Generation zu homogenen Populationen führen, entgegenwirken.

Das Heiratsschema legt fest, welche Individuen einer Population zur Erzeugung von Nachkommen und damit neuer Chromosomen herangezogen werden. Klassisch ist die Wahrscheinlichkeit der Auswahl der Kandidaten proportional zu ihrer Fitness. Damit sind hochbewertete Elemente in der nächsten Generation mit höherer Wahrscheinlichkeit vertreten als durchschnittliche. Das Ziel ist ein kontinuierliches Ansteigen der Güte der Population. Ein Beispiel ist das Heiratsschema "Roulette".[SHF94]

Nachdem mit Hilfe des Heiratsschemas neue Nachkommen erzeugt wurden, muss entschieden werden, was mit den bisherigen Individuen der Population gesche-

hen soll. Diese Aufgabe übernimmt das Ersetzungsschema. Die einfachste Form ist das "generational replacement" Schema, bei dem die aktuelle Population vollständig durch ihre Nachkommen ersetzt wird. Dies wird auch als nichtüberlappende Population bezeichnet. Die Gefahr dabei ist, dass das beste Individuum der neuen Generation schlechter als ein durchschnittliches der Elterngeneration ist. Bei der Anwendung des Prinzips der Eliten werden die n besten Individuen in die nächste Generation übernommen. Unter Umständen kommt es zu einer Dominanz bei großen Unterschieden in der Güte (Stichwort "Superindividuum") und damit zu einer Stagnation der Suche. Um diesen Effekt zu reduzieren, gibt es verschiedene Mechanismen (siehe [SHF94]).

Üblicherweise wird die Gesamtgröße der Population konstant gehalten. Es werden n Nachkommen erzeugt und anschließend die n schlechtesten Individuen der Population entfernt. Die Populationen der einzelnen Generationen überlappen sich. Ist n viel kleiner als die Populationsgröße, so wird dies als "steady-state" Ersetzungsschema bezeichnet. Ist n gleich der Populationsgröße, so tritt "generational replacement" auf. Dazwischen existieren viele verschiedene Variationen.[SHF94]

Zusammenfassend lässt sich feststellen, dass zwischen ES und GA teilweise gravierende Unterschiede bestehen. Bei ES bedeutet *survival of the fittest* im strengen Sinne, dass die μ besten Individuen mit gleicher Wahrscheinlichkeit die neuen Eltern werden. Bei GA steigt die Wahrscheinlichkeit der Auswahl proportional mit der Fitness. Durch die bestehenden Unterschiede lassen sich beide Modelle nur schwer vergleichen. Sie sind stark von ihren Parametereinstellungen abhängig, welche nur schwer optimal abschätzbar sind. Zur Beurteilung der EA werden daher Testfunktionen herangezogen, die jedes Verfahren gleichermaßen bevorzugen beziehungsweise benachteiligen. Sie testen unter anderem, wie leicht die Algorithmen in lokalen Optima steckenbleiben. Die Testfunktionen und der jeweilige Vergleich zwischen den Algorithmen findet sich in [SHF94], ebenso wie einige Punkte zur Parallelisierbarkeit, auf die an dieser Stelle nicht näher eingangen werden soll.

In dieser Arbeit wird das Modell "Genetische Algorithmen" in Form der Bibliothek GAlib eingesetzt.

2.8.5 Die Bibliothek GAlib

GAlib ist eine, für nichtkommerzielle Zwecke freie, C++-Bibliothek des MIT zur Umsetzung von Genetischen Algorithmen. Sie enthält vorgefertigte Strukturen und Vorlagen, welche für spezifische Zwecke modifiziert werden können. Die Bibliothek ist objektorientiert umgesetzt.

Es werden sowohl überlappende (Steady-State GA) als auch nichtüberlappende (Simple GA) Populationen unterstützt. Der Umfang der Überlappung kann angepasst werden. Für andere Genetische Algorithmen sind Bespiele vorhanden. Neue können leicht durch Nutzung der vorhandenen Strukturen erstellt werden.

Die Parameter für die Genetischen Algorithmen können über Dateien, die Kommandozeile und/oder den Quellkode konfiguriert werden. Eine Parallelverarbeitung wird mit PVM (Parallele Virtuelle Maschine) ermöglicht.

Die Bibliothek hat vorgefertigte Endkriterien wie beispielsweise Konvergenz und Anzahl der Generationen. Die Artenbildung kann entweder mit einer Ersetzungsstrategie (DeJong crowding) oder durch eine Fitnessskalierung (Goldberg sharing) umgesetzt werden.

Für überlappende Populationen sind Ersetzungsstrategien vorhanden. Diese umfassen Elternersetzung, zufällige Ersetzung und Ersetzung des schlechtesten Individuums. Elitarismus ist für nichtüberlappende Populationen optional. Die eingebauten Selektionsmethoden umfassen unter anderem Rang, Roulettrad, Turnier und zufällige Auswahl. Der Ersetzungs- und der Selektionsoperator sind anpassbar.

Statistiken für den GA, beispielsweise für die Bewertung, werden automatisch generiert und können abgespeichert werden.

Die Chromosomen können aus jedem C++-Datentyp bestehen. In dieser Arbeit werden eindimensionale Felder mit Fließkommazahlen verwendet. Die Initialisierung, Mutation, Kreuzung und der Vergleich der Chromosomen kann angepasst werden, es stehen allerdings auch vorgefertigte Operatoren zur Verfügung.[Wal96]

Eine ausführlichere Darstellung aller Funktionen findet sich in der Dokumentation [Wal96] der Bibliothek.

Der Vorgehensweise bei der Verwendung der Bibliothek gliedert sich in drei Schritte:

- Repräsentation/Chromoson definieren
- Genetische Operatoren definieren
- Zielfunktion (Bewertungsfunktion) setzen.

Die Repräsentation einer Lösung des Problems muss in einer Datenstruktur erfolgen können. Der Genetische Algorithmus erzeugt eine Population von Lösungen auf Basis dieser und operiert dann auf dieser um die beste Lösung zu finden. Die Bibliothek bringt vier verschiedene Chromosomentypen für binäre, zeichenketten- und fließkommabasierte Kodierungen mit.

Es gibt, wie in Abschnitt 2.8 dargestellt, unzählige Varianten von GA. GAlib beinhaltet die drei Basistypen `simple`, `steady-state` und `incremental`. Sie unterscheiden sich in der Art der Erzeugung neuer Individuen und wie sie alte im Laufe der Generation ersetzen. Die Bibliothek unterstützt auch parallele GA.

Der allgemeine Ablauf eines GA ist in Abbildung 2.12 dargestellt. Nach der Initialisierung der Population werden Individuen für Paarung und Bildung von Nachkommen ausgwählt. Die daraus entstehenden Kinder werden anschließend entsprechend der Mutationsrate mutiert und in die Population eingefügt. Dieser Ablauf wird wiederholt, bis das Stoppkriterium erreicht ist. Dieses muss festgelegt werden, beispielsweise durch die Anzahl der Generationen, der Konvergenz der Population oder durch einen problemspezifischen Ansatz.

Die GA und ihre Eigenschaften sind in Abschnitt 2.8 näher erklärt, an dieser Stelle soll nur eine kurze Zusammenfassung erfolgen. Der `simple` GA nach Goldberg

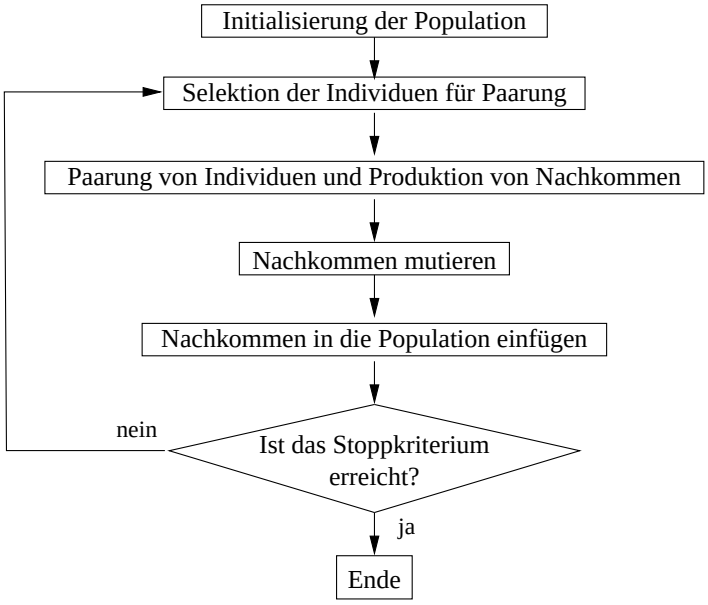


Abbildung 2.12: Allgemeiner Ablauf eines Genetischen Algorithmus

arbeitet mit nichtüberlappenden Populationen. Das heißt, dass in jeder Generation eine komplett neue Population von Individuen erzeugt wird. Optional bietet die Bibliothek Elitarismus. Überlappende Populationen werden durch den `steady-state` GA realisiert. Die Ersetzungsrate gibt an, wie viele Individuen in der nächsten Generation ersetzt werden. Beim `incremental` GA besteht eine Generation nur aus 2-3 Kindern. Angepasste Ersetzungmethoden definieren, wie sie in die bestehende Population integriert werden.[Wal96]

Die Repräsentation soll minimal, aber umfassend sein. Sie muss alle Lösungen repäsentieren. Wenn unpassende Lösungen entstehen können, muss die Bewertungsfunktion dies berücksichtigen. Die Bewertungsfunktion bestimmt, welches Individuum besser als ein anderes ist.

Jedes Genom besitzt die drei primären Operatoren Initialisierung, Mutation und Kreuzung. Alle können angepasst werden. Die Initialisierung füllt die Chromosomen mit dem ursprünglichen Material, aus dem sich alle Lösungen entwickeln werden.

Der Mutationsoperator legt die Mutationsprozedur für jedes Genom fest. Sie sollte sowohl in der Lage sein, neues genetisches Material einzuführen, als auch existierendes zu verändern.

Der Kreuzungsoperator definiert die Prozedur, wie ein Kind aus den Elterngenomen zu generieren ist.

Neben der Bewertungsfunktion existiert ein Vergleichsoperator, welcher von manchen GA benötigt wird. Er gibt an, in wieweit sich ein Genom von einem anderen unterscheidet.

Das Populationsobjekt ist ein Kontainer für Genome. Es protokolliert die besten Individuen, die durchschnittliche Abweichung der Population und beinhaltet

Selektionsmethoden, welche die Partner für die Fortpflanzung auswählen.

Das Skalierungsschema wandelt die Bewertungspunkte jedes Individuums in einen Fitnesswert, welchen der GA für die Selektion, und damit für die Fortpflanzung nutzt (siehe Abschnitt 2.8). Jede Transformation der Bewertung in den Fitnesswert ist möglich, typischerweise ist sie linear.

Für die konkrete Umsetzung und Bespiele ist die Dokumentation der Bibliothek [Wal96] zur Rate zu ziehen.

2.9 Neuronale Netze

Neuronale Netze orientieren sich in ihrem Aufbau und ihrer Konzeption an der Funktionsweise des menschlichen Gehirns. Sie können deshalb dazu genutzt werden, wichtige geistige Fähigkeiten des Menschen nachzubilden und zu simulieren. Dazu gehören beispielsweise das Lernen aus Beispielen, das schnelle Erkennen und Vervollständigen komplizierter Muster, sowie das assoziative Speichern und Abrufen von Informationen.

Das menschliche Gehirn kann innerhalb von Sekundenbruchteilen kompliziert Signale erkennen und inhaltlich interpretieren, selbst wenn diese unbekannt, das heißt, zuvor noch nie erfasst worden sind. Jeder Rechner mit klassischer Von-Neumann-Architektur[1] wäre trotz höherer Datenverarbeitungsgeschwindigkeit und Genauigkeit dazu nur bedingt in der Lage. Die Lösung liegt in der parallelen Informationsverarbeitung im Gehirn.

Das Gehirn enthält bei einem durchschnittlichen Gewicht von 1500 Gramm zwischen 10 und 100 Milliarden Nervenzellen (Neuronen). Die Verarbeitungskapazität eines einzelnen Neurons ist verglichen mit heutigen Computern gering. Da die Nervenzellen aber untereinander hochgradig vernetzt sind und miteinander kommunizieren, ergibt sich eine erheblich größere resultierende Kapazität. [SHG90]

Im folgenden Abschnitt werden die mathematischen Grundlagen für Neuronale Netze erläutert. Anschließend folgen generelle Aussagen über das Lernen von Funktionen und die damit verbundene Konvergenz. Abschnitt 2.9.3 zeigt verschiedene Klassifizierungsmöglichen Neuronaler Netze auf und Abschnitt 2.9.4 geht näher auf das in dieser Arbeit verwendete Modell ein. Darauf folgen generelle Anwendungsmöglichenkeiten. Die in der vorliegenden Arbeit eingesetzte Bibliothek `FANN` wird in Abschnitt 2.9.6 zusammengefasst. Abschließend werden einige Anwendungen von Neuronalen Netzen in der Mikrosystemtechnik aufgeführt.

Für Begriffe im Zusammenhang mit Neuronalen Netzen gibt VDI/VDI 3550 [Nor03] Richtlinien vor.

[1] eine zentrale Verarbeitungseinheit (CPU) und Arbeitsspeicher

2.9.1 Mathematische Grundlagen

Ähnlich wie das menschliche Gehirn aus Gehirnzellen aufgebaut ist, besteht ein neuronales Netz aus elementaren Verarbeitungseinheiten, den Neuronen. Diese Neuronen stehen untereinander in Verbindung und beeinflussen sich gegenseitig.

Das formale Neuron hat, wie in Abbildung 2.13 dargestellt, eine Eingabemenge I (Input) und eine Ausgabemenge O (Output).

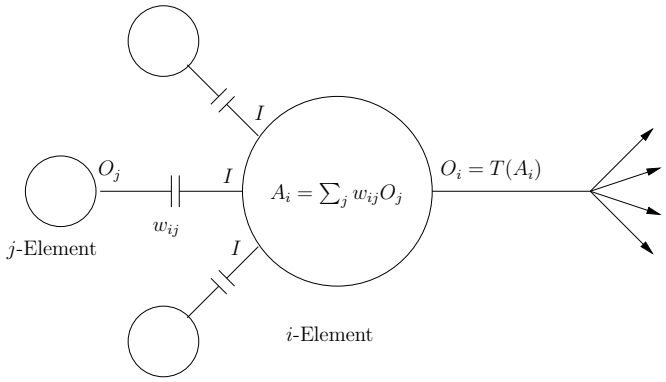


Abbildung 2.13: Das formale Neuron

Der Input I eines Neurons E_i lässt sich als Vektor interpretieren. Seine Komponenten lassen sich aus dem Output O_j der Elemente E_j, die mit E_i eine Verbindung haben, und den Gewichten w_{ij} dieser Verbindung berechnen. Der Output O_i ist einzelner, meist reeller Wert, welcher durch die Ausgabefunktion f_i aus der momentanen Aktivierung $a_i(t)$ berechnet wird.[SHF94]
Es gilt:

$$f_i(a_i(t)) = O_i(t)$$

Die Gewichte w_{ij} beeinflussen das Signal O_j. Sind sie kleiner als 1 wirken sie hemmend, ansonsten verstärkend.[Maz92]

Anhand der Outputs der E_j, die einen Input zu E_i liefern, berechnet die Propagierungsfunktion (Inputfunktion) NET_i den entsprechenden Input net_i. Dieser wird auch gewichteter Input genannt.

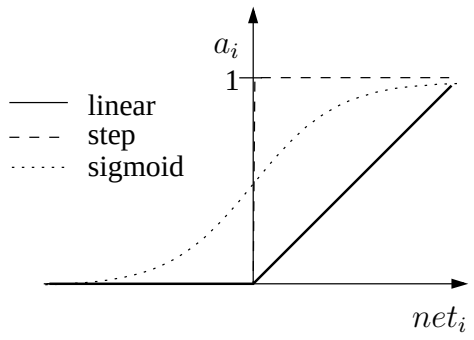
$$NET_i(O_j(t), w_{ij}) = net_i(t)$$

Gebräuchliche Inputfunktionen net_i ergeben sich aus der Summation oder aus dem Produkt der gewichteten Eingaben $w_{ij} \cdot O_j$, weitere sind in [SHF94] aufgeführt.

Die Aktivierungsfunktion oder auch Transferfunktion F_i berechnet anschließend aus dem bisherigen Aktivierungszustand $a_i(t)$ und dem aktuellen gewichteten Input $net_i(t)$ den neuen Aktivierungszustand $a_i(t+1)$ für jedes Element E_i.[SHF94]

$$a_i(t+1) = F_i(a_i(t), net_i(t))$$

Ausgewählte typische Transferfunktionen sind in Abbildung 2.14 dargestellt, weitere sind in [SHF94] zu finden.



linear $a_i = net_i$

step $a_i = \begin{cases} 1 & \text{wenn } net_i > 0 \\ 0 & \text{sonst} \end{cases}$

sigmoid $a_i = \frac{1}{1+exp(-g \cdot net_i)}$

net_i ... gewichteter Input des Elements E_i

a_i ... Aktivität von E_i

g ... Parameter für die Steilheit der Kurve

Abbildung 2.14: Ausgewählte Transferfunktionen

Aus dem aktuellen Aktivierungszustand $a_i(t)$ wird der Output des Elements E_i über die bereits erwähnte Outputfunktion f_i berechnet. Dies kann direkt geschehen mit

$$O_i = a_i$$

oder als winner-take-all Funktion, bei dem das Element k mit der höchsten Aktivität gewinnt, wobei O_i eine beliebige Funktion ist.[SHF94]

$$O_k = \begin{cases} O_i(a_k) & \text{wenn } a_k = max(a_i) \\ 0 & \text{sonst} \end{cases}$$

Der Informationsfluss in einem Neuron wird in Abbildung 2.15 verdeutlicht. [SHF94]

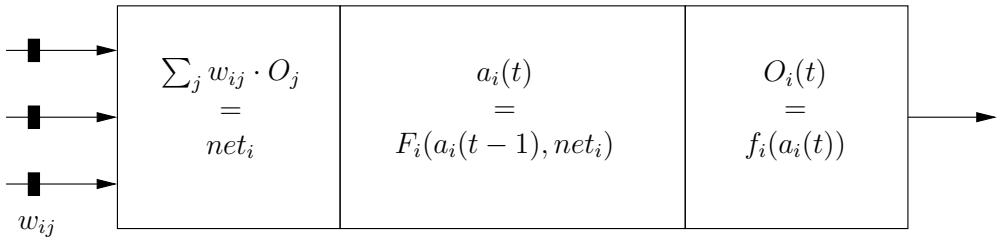


Abbildung 2.15: Informationsfluss in einem Element i

Der Funktionsablauf in einem Neuron ist in Abbildung 2.16 zusammengefasst.

Ein neuronales Netz wird durch folgende Komponenten charakterisiert [SHF94]:

- Die Menge der Neuronen $E = \{E_j | j <= N\}$, wobei N die Anzahl der Neuronen im Netz angibt.
- Die Menge der Aktivierungszustände $A = \{a_i(t) | i <= N, t < \infty\}$, welche sich mit der Zeit verändern können (z. B. beim Lernen).
- Die Netzwerktopologie, das heißt die Verbindungen zwischen den Elementen, welche durch Gewichtsmatrizen dargestellt werden. Die Notation w_{ij} kennzeichnet das Gewicht der Verbindung von E_j (Sender) zu E_i (Empfänger).

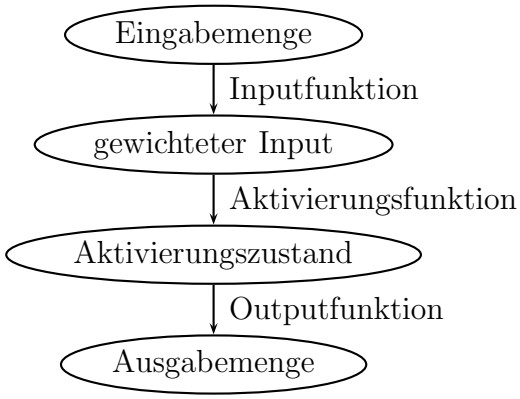


Abbildung 2.16: Funktionsablauf in einem Neuron

- Eingabemenge I und Ausgabemenge O der einzelnen Elemente E_i.
- Die Propagierungs- oder Inputfunktion NET_i
- Die Aktivierungsfunktion F_i
- Die Ausgabefunktion f_i
- Die Lernregel Δw_{ij}

Die Lernregel Δw_{ij} beschreibt Veränderungen der Gewichte zwischen den Neuronen nach Gleichung 2.9.

$$\Delta w_{ij} = w_{ij}(t+1) - w_{ij}(t) \tag{2.9}$$

Die Veränderung der Gewichte ist abhängig von dem tatsächlichen Aktivierungszustand der Elemente $a_i(t)$, dem erwarteten Aktivierungszustand oder Output $z_i(t)$, dem gewichteten Input net_i und den alten Gewichten zwischen den Elementen. Die Lernregel lautet in allgemeiner Form

$$\Delta w_{ij} = g(a_i(t), z_i(t)) \cdot h(o_j(t), w_{ij}) \tag{2.10}$$

wobei $z_i(t)$ für den erwartenen Aktivierungszustand steht. Bei einer konkreten Lernregel müssen nicht alle Parameter berücksichtigt werden. Sie ist vom Netzmodell abhängig. Durch Einsetzen der Gleichungen 2.11 und 2.12 in die

$$g(a_i(t), z_i(t)) = \sigma \cdot (z_i(t) - a_i(t)) \tag{2.11}$$

$$h(o_j(t), w_{ij}) = o_j(t) \tag{2.12}$$

allgemeine Form (Gleichung 2.10) ergibt sich die klassische Widrow-Hoff-Lernregel (Gleichung 2.13), welche auch als Delta-Lernregel bezeichnet wird. [SHF94]

$$\Delta w_{ij} = \sigma(z_i(t) - a_i(t)) \cdot o_j(t) \tag{2.13}$$

Die Lernrate σ ist eine reelle Zahl zwischen 0 und 1 und gibt an, wie stark ein einzelner Lernschritt in die Veränderung der Gewichte eingeht. Um gelernte Muster wieder zu löschen, kann die Lernrate auch negativ sein.[SHF94]

Die Gewichte werden bei der Delta-Lernregel so verändert, dass der Fehler δ_i für ein Element einen vorgegebenen Wert unterschreitet. Er errechnet sich aus der Differenz zwischen dem Zieloutput z_i und der Aktivität a_i (Gleichung 2.14)

$$\delta_i(t) = z_i(t) - a_i(t) \tag{2.14}$$

Die Anpassung der Gewichte erfolgt dann für die n Eingänge des Elementes i nach Gleichung 2.15.

$$w_{ij}(t+1) = w_{ij}(t) + \frac{\sigma}{n}\delta_i o_j \tag{2.15}$$

Jedes Element lernt bis der gewünschte Fehler erreicht ist. Auch Elemente mit einer richtigen Ausgabe können ihre Gewichte justieren. Ihre Aussage wird damit „sicherer“ und dadurch die Stabilität des gelernten Musters größer. Das Netz „vergisst“ beim Lernen weiterer Muster alte Muster weniger schnell. Zusätzlich konvergiert das Netz schneller. Ein Lernschritt wird als *Epoche* bezeichnet.

Die Delta-Regel ist eine sehr häufig verwendete Lernregel und Basis für weitere Entwicklungen.[SHF94]

2.9.2 Berechenbarkeit, Lernen und Eigenschaften

Ein Neuronales Netz ist ein dynamisches System, welches in der Lage ist, eine beliebige, nicht analytisch beschriebene, Funktion $y = f(x)$ auszuführen. Seine Struktur ist durch die Anzahl der Elemente, seine Topologie, auf welche in Abschnitt 2.9.3 näher eingegangen wird, und die Funktionen in den Neuronen definiert.

Die Informationen des Neuronalen Netzes liegen in den Aktivierungen der Neuronen. Die Wissensbasis umfasst zusätzlich die Gewichte der Verbindungen. Die Gewichte werden in der Lernphase eingestellt. Die Verarbeitung erfolgt durch die Neuronen kollektiv und verteilt. Die Vorteile Neuronaler Netze liegen in der Verallgemeinerung und Assoziation von Daten. Sie sind im Vergleich zu analytischen Verfahren unempfindlicher gegenüber Störungen und Defekten. Durch Parallelisierung kann die Verarbeitungsgeschwindigkeit erhöht werden.[Maz92]

Die Nachteile liegen in den im Vergleich zum menschlichen Gehirn mangelhaften logischen Fähigkeiten und der niedrigen Genauigkeit. Weiterhin kann ein Neuronales Netz keine Erklärungen zu den Ergebnissen liefern, das heißt die Interpretation muss durch den Menschen erfolgen.[Maz92]

Die Selbstanpassung innerhalb eines Neuronalen Netzes auf ein gewünschtes Verhalten wird als „Lernen“ bezeichnet. Diese Lernfähigkeit entbindet von der expliziten Suche nach einer algorithmischen Lösung des gestellten Problemes.[SHF94]

Das allgemeine Neuronale Netz in Abbildung 2.17 berechnet eine Funktion

$$y = f(x)$$

wobei x die Eingabe und y die Ausgabe des Netzes sind. Am Anfang sind alle Elemente inaktiv und das Netz hat eine Ausgabe gleich Null. Werden von außen

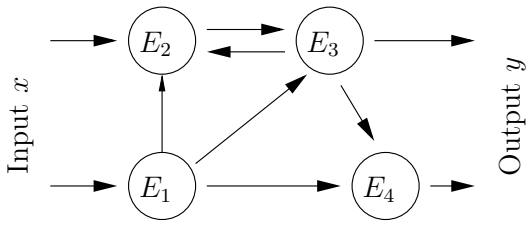


Abbildung 2.17: Allgemeines Neuronales Netz

Eingabesignale x angelegt, werden die Inputelemente angesprochen und reizen weitere Elemente. Jedes Element des Netzes wird aktiviert und damit die Ausgabe y des Netzes gebildet.

Da Neuronale Netze zyklische Verbindungen haben können — wie in Abbildung 2.17 zwischen den Elementen $E2$ und $E3$ — werden sich in diesen Fällen einige Elemente gegenseitig reizen. Dabei ergeben sich drei Fälle:

1. Die gegenseitige Stimulierung führt zur Aktivierung der Outputelemente mit wechselnden Werten, damit hat das Netz keine Konvergenz. Im Falle einer stetig wachsenden Stimulierung explodiert das Netz.[Maz92]

2. Die gegenseitige Stimulierung führt zu einer sich zyklisch wiederholenden Sequenz von Aktivierungen. Der Output schwingt.[Maz92]

3. Die gegenseitige Stimulierung führt zu einer sich nicht mehr ändernden Kombination von Aktivierungen, das Netz strebt zu einer Lösung $y = f(x)$ und ist damit eine deterministische Maschine. Es existieren Theoreme, welche die Grundvoraussetzung für konvergente Netze festlegen.[Maz92]

Das Neuronale Netz lernt durch ein Training anhand von vorgestellten Beispielen. Es baut die nötigen internen Kenntnisse, das heißt die Einstellung der Gewichte, selbst auf, um die geforderte Aufgabe zu erfüllen.

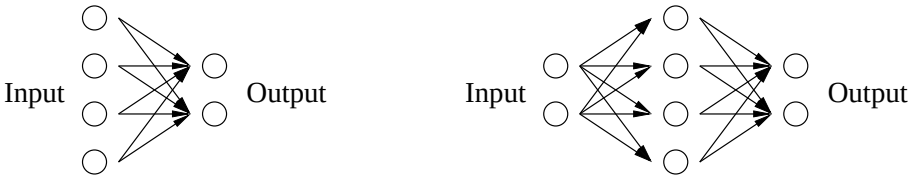
Die Lernfähigkeit ist das Ergebnis einer sorgfältigen Auswahl der Netztopologie, der Anzahl der Elemente, der Lernregel und der Menge an Beispielen, sowie Input-, Output- und Aktivierungsfunktion. Bei einem nicht angemessenen Aufbau kann das Neuronale Netz unter Umständen nicht in der Lage sein, die gewünschte Funktion abzubilden.

Es werden zwei Lernmethoden unterschieden: assoziativ (mit Unterweisung) und entdeckend (ohne Unterweisung). Das assoziative Lernen erfolgt durch Trainingsdatensätze, welche dem Netz sowohl Input- als auch Output vorgeben. Ein verfälschter Eingabevektor führt dann beim Ausführen des Neuronalen Netzes im Idealfall zur Ausgabe des unverfälschten originalen Outputvektors [SHF94]. Ein Anwendungsbeispiel wäre die Gesichtserkennung. Das entdeckende Lernen erfordert nur den Inputvektor. Es geht dabei darum, die beobachteten Beispiele zu organisieren und einzuordnen und dabei Beziehungen und Regelmäßigkeiten zu entdecken.[Maz92]

2.9.3 Klassifizierung Neuronaler Netze

Neuronale Netze lassen sich anhand verschiedener Parameter einteilen. Mögliche Kriterien sind beispielsweise:

- Zahl der Schichten, wie in Abbildung 2.18 dargestellt
- Kopplung zwischen den Neuronen
- Lernregel
- Akzeptanz von binären oder stetigen Eingaben



(a) Einschichtiges Netz (b) Mehrschichtiges Netz

Abbildung 2.18: Klassifizierung Neuronaler Netze nach Anzahl der Schichten

Eine weitere mögliche Einordnung ist in Abbildung 2.19 (Quelle: [SHF94]) dargestellt. Feed-Forward-Netzwerke sind rückkopplungsfreie Netzwerke, in denen der

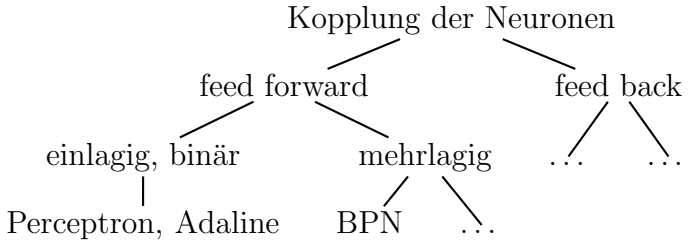


Abbildung 2.19: Klassifizierung von neuronalen Netzwerken nach Modellen

Informationsfluss nur in eine Richtung geht. Es gibt keine Verbindungen von einem Neuron einer höheren Schicht, zu einem einer niedrigeren Schicht. Diese Art von Netzwerk spaltet sich weiter in ein- und mehrschichtige Netze auf.[SHF94]

Perceptron und Adaline waren die Vorläufer des späteren Neuronalen Netze, ihre Leistungsfähigkeit ist beschränkt.[SHF94]
Das Backpropagation Netzwerk (BPN) wird im folgenden Abschnitt kurz beschrieben. Es wird wegen seiner Vielseitigkeit in dieser Arbeit eingesetzt. Auf Grund der zahlreichen Varianten Neuronaler Netze, wird an dieser Stelle nicht auf weitere Spezialformen eingegangen. Ausführlichere Darstellungen finden sich in [SHF94], [Maz92] und [Che93].

Die Kategorie der Feedback-Netzwerke umfasst die rückgekoppelten Netzwerkei, in denen der Informationsfluss nicht festgelegt ist. Die Schichten sind nicht

hierarchisch geordnet und der Output jedes Elementes kann jedem anderen als Input dienen. [SHF94]

Im Gegensatz zu rückkopplungsfreien Netzen, welche eine definierte, eindeutige Ausgabe auf jede Eingabe zeigen, liefern rückgekoppelte Netze erst dann eine eindeutige Ausgabe, wenn sie nach endlich vielen Durchläufen einen stabilen Zustand erreichen. Dieser Zustand liegt vor, wenn sich der erzeugte Output aller Neuronen als Eingabe in das Netz gespeist, selbst reproduziert.[SHF94] Diese Kategorie ist nur der Vollständigkeit halber erwähnt und wird nicht weiter betrachtet.

Eine ausführliche Beschreibung konkreter gebräuchlicher Neuronaler Netze findet sich in [SHF94].

2.9.4 Das Backpropagation Netzwerk

Für die Leistung und Repräsentationsmöglichkeiten von rückkopplungsfreien Netzwerken ist die Zahl der Schichten mit variablen, lernenden Gewichten von entscheidender Bedeutung. Die Gewichte, der dabei nicht von außen zugänglichen, also verdeckten Schichten, müssen in geeigneter Weise angepasst werden. Der Backpropagationalgorithmus setzt diese Anpassungen für beliebige, rückkopplungsfreie Netzwerkstrukturen um.

Die Delta-Lernregel aus Abschnitt 2.9.1 berechnet einen Fehler für jedes Outputelement. Die Anpassung der Gewichte erfolgt proportional zu diesem Fehler. Die generalisierte Delta-Lernregel im BPN ermöglicht die Berechnung von Fehlern auch für verdeckte Schichten. Der Fehler einer höheren Schicht wird dabei den Elementen der vorhergehenden Schicht zurückpropagiert und ihnen zugeschrieben. Mit diesem Vorgehen können beliebig viele Schichten adaptiert werden.

Ein BPN besteht aus einer Eingabe- und einer Ausgabeschicht. Zwischen diesen liegen verdeckte Schichten. Das Ausführen entspricht dem jedes anderen Feed-Forward-Netzwerkes. Die Neuronen berechnen den Output nach Gleichung 2.16 aus dem angelegten Input-Vektor.

$$o_i = a_i = F\left(\sum_j (w_{ij} o_j)\right) \tag{2.16}$$

Der Fehler des gesamten Netzes wird nun nicht durch einfache Differenzbildung, sondern aus der Summe der Quadrate der Differenzen berechnet (Gleichung 2.17).

$$E = \frac{1}{2}\sum_i (z_i - o_i)^2 \tag{2.17}$$

Die Minimierung dieses Fehlers erfolgt durch Adaption der Gewichte in Richtung des steilsten Gradienten. Für alle Elemente muss gelten:

$$w_{ij}(t+1) = w_{ij}(t) + \sigma \delta_i o_j \tag{2.18}$$

Der Fehler δ_i für verdeckte Neuronen ist

$$\delta_i = F'(\mathrm{net}_i)\sum_h (\delta_h w_{hi}) \tag{2.19}$$

δ_h bezeichnet dabei den Fehler der Elemente, die vom Eingang aus gesehen, in der nächsten Schicht hinter der Schicht i liegen. Das Gewicht w_{hi} ist die Verbindung zu diesen Elementen. Die Lernregel ist rekursiv, für die Berechnung des Fehlers in einer Schicht müssen die entsprechenden Fehler in der darauffolgenden Schicht bekannt sein. Der Fehler der Ausgabeschicht ist

$$\delta_i = F'(\text{net}_i) \cdot (z_i - o_i) \tag{2.20}$$

Die Transferfunktion F muss differentierbar sein. Die am häufigsten eingesetzte Transferfunktion ist die Sigmoidfunktion, welche in Abschnitt 2.9.1 bereits erwähnt wurde. Die Ableitung der Sigmoidfunktion lautet

$$F'(\text{net}) = \frac{\partial F(\text{net})}{\partial \text{net}} = F(\text{net}) \cdot (1 - F(\text{net})) \tag{2.21}$$

Weitere häufig eingesetzte Transferfunktionen des BPN sind die Sinusfunktion und der Tangens Hyperbolicus.

Die Initialisierung der Gewichte erfolgt durch kleine zufällige Werte. Dies verhindert die sonst entstehenden Symmetrien in den verdeckten Schichten, welche zu immer gleichen Gewichtsänderungen führen würden. Für Probleme, die unterschiedliche Gewichte in diesen Schichten verlangen, könnte das Netz nicht konvergieren.

Um die Lerngeschwindigkeit zu erhöhen, wird ein Momentumfaktor μ eingeführt, mit dem zusätzlich die Gewichtsveränderung des vorherigen Lernschrittes — und damit dessen Fehler — eingeht. Die Gewichtsberechnung erfolgt somit nach Gleichung 2.22

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \Delta w_{ij}(t) \\ \text{mit} \quad \Delta w_{ij}(t) &= \sigma \delta_i o_j + \mu \Delta w_{ij}(t-1) \end{aligned} \tag{2.22}$$

Der Momentumfaktor liegt zwischen 0 und 1, typischerweise wird er auf einen hohen Wert gesetzt (z. B. $0,9$). Die Lernrate sollte $0,5$ betragen.

Durch diese Art der Gewichtsänderung werden lokale Minima für die aktuelle Gewichtsänderung nicht so stark berücksichtigt. Das Momentum wirkt wie eine Art Tiefpassfilter.

Die Lernprozedur eines BPN entspricht dem sogannten Hill-Climbing-Algorithmus. Dessen Aufgabe ist es, die Stelle mit dem kleinsten Fehler, also das tiefste Tal des Fehlergebirges, zu finden. In Abhängigkeit seiner Form besteht die Gefahr, in ein lokales Minimum zu geraten, welches auf Grund der Fehlerfunktion nicht mehr verlassen werden kann. In der Regel ist es aber nicht möglich, die Form des Fehlergebirges für ein Problem voraus zu sagen und damit auf die Lösbarkeit und Konvergenz zu schließen. Nehmen die Gewichte in einem Bereich des Netzwerkes betragsmäßig sehr große Werte an, so kann das Lernen zum Stillstand kommen, da die Ableitung der Sigmoidfunktion für solche Werte fast 0 ist. Dem kann mit einer niedrigen Lernrate und niedrigem Momentumfaktor entgegengewirkt werden. Die Lernzeit wird mit dieser Maßnahme jedoch auch verlängert.[SHF94]

2.9.5 Anwendungen

Neben zahlreichen Anwendungsmöglichkeiten, wie Prognosen, Analysen und Optimierungen, welche in [SHF94] und [Che93] näher beschrieben werden, können mit Neuronalen Netzen Muster erkannt werden. Diese Fähigkeit findet in dieser Arbeit in der Eigenmodenerkennung von Siliziumteststrukturen in Form eines Back-Propagation-Modelles ihre Verwendung. Die Umsetzung wird über die Open-Source-Bibliothek Fast Artificial Neural Network (FANN) realisiert, welche im folgenden Abschnitt kurz vorgestellt wird.

2.9.6 Die Bibliothek FANN

Die FANN-Bibliothek ist in der Programmiersprache C implementiert und setzt mehrschichtige Neuronale Netze des BPN-Modelles um. Die Benutzung ist kompakt, gut dokumentiert und erlaubt die Einstellung zahlreicher Parameter. Verschiedene Transferfunktionen sind vorhanden. Zusätzlich verfügt die Bibliothek über Schnittstellen zu zahlreichen Programmiersprachen und Programmen, wie C++, .NET, Mathematica, MathLab und Octave. [Nis09]

Die Bibliothek stellt Komponenten zur Trainings- und Ausführungsphase des Neuronalen Netzes bereit. Die grundlegenden Komponenten werden anschließend näher erklärt. Über Details gibt [Nis03] Aufschluss. Die Bibliothek ist auf Schnelligkeit ausgelegt. Obwohl es genauere Algorithmen als den BPN gibt wurde dieser ausgewählt, da er einfach und in den meisten Fällen effektiv genug ist. Die konkrete Implementierung und die verfügbaren Funktionen können aus [Nis03] entnommen werden.

Wie in Abschnitt 2.9 bereits erwähnt, kann ein Verwendung eines Neuronalen Netzes in eine Trainings- und eine Ausführungsphase unterteilt werden. Die Bibliothek FANN realisiert die beiden Phasen mit mehreren Komponenten, wie in Abbildung 2.20 dargestellt. Die Trainingsphase gliedert sich in den Aufbau ei-

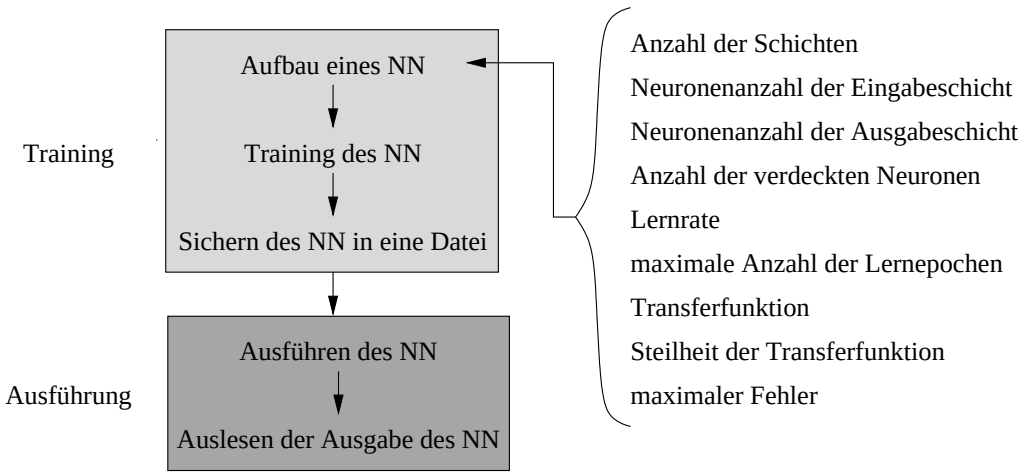


Abbildung 2.20: Komponenten der FANN-Bibliothek

nes Neuronalen Netzes, das Lernen und das Sichern des Netzes in eine Datei. Als einstellbare Parameter seien hier die Anzahl der Schichten, die Lernrate, die Transferfunktion und ihre Steilheit, sowie die Anzahl der Anpassungen (Epoche) und das Abbruchkriterium (der Fehler des Neuronalen Netzes) genannt. Zusätzlich fließt die Anzahl der Ein- und Ausgaben ein. Beim Lernen des Neuronalen Netzes liefert die Bibliothek eine Reportausgabe über die Anzahl der Epochen und den aktuellen Fehler des Netzes. Der Fehler des Netzes ist der Mittelwert der Fehler der Outputneuronen. Die programmiertechnische Ausgabeumleitung der Reportausgabe in eine Datei ist nicht ohne weiteres möglich. Die Ursache liegt in der Implementierung der Bibliothek. Diese Ausgaben werden stets auf der Standardausgabe angezeigt. Die manuelle Ausgabeumleitung kann mittels „>“ erfolgen.

Die Ausführungsphase beinhaltet das Ausführen des gespeicherten Neuronalen Netzes mit Eingabedaten und das Auslesen der zugehörigen Ausgabe.

In der Anleitung wird empfohlen, die Trainings- und Ausführungsphase jeweils in unterschiedlichen Programmen zu realisieren, da im Allgemeinen ein Neuronales Netz einmal trainiert und danach mehrfach ausgeführt wird. [Nis03]

Die Ein- und Ausgabedaten für das Neuronales Netz (NN) sind über Dateien bereitzustellen. Dabei ist eine Formatierung, wie in Listing 2.1 abgebildet, einzuhalten.

Listing 2.1: FANN Datenformatierung

```
<Anzahl Datenpaare (DP) j> <Anzahl Inputs i> <Anzahl Outputs o>
<Datenpaar 1 Input 0> <DP 1 Input 1> ... <DP 1 Input i>
<Datenpaar 1 Output 0> <DP 1 Output 1> ... <DP 1 Output o>
...
<Datenpaar j Input 0> <DP j Input 1> ... <DP j Input i>
<Datenpaar j Output 0> <DP j Output 1> ... <DP j Output o>
```

Die erste Zeile der Trainingsdatei enthält leerzeichengetrennt die Anzahl der Datenpaare j, die Anzahl der Inputs i sowie die Anzahl der Outputs o. In den Zeilen 2 bis $j+1$ folgen die Datenpaare. Ein Datenpaar j besteht aus i Inputs und o Outputs, welche durch einen Zeilenumbruch getrennt sind. Der Datentrenner ist das Leerzeichen. In Abbildung 2.21 ist die Trainingsdatei der `XOR`-Funktion dargestellt. Die Bereitstellung der Inputdaten für die Ausführung kann direkt im Programm erfolgen. Das Lesen aus einer Datei muss zusätzlich realisiert werden. Die Anzahl der Inputs der Ausführung muss gleich der aus den Trainingsdaten sein. Beim Ausführen des NN mit diesen Daten liefert es den entsprechenden Output, welcher ausgelesen werden muss. Die Implementierung zum Lesen und Ausführen eines NN für die `XOR`-Funktion ist in [Nis03] nachzulesen.

Die Einstellung eines NN hängt stark von der Anwendung ab. Die Parameter resultieren entweder aus der Optimierung des Netzes oder aus Erfahrung. Der Autor schlägt einige Werte vor, die seiner Meinung nach in den meisten Fällen eine günstige Wahl darstellen.

Die Lernrate ist ein sehr wichtiger Parameter, allerdings schwer zu schätzen.

Wertetabelle:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Trainingsdatei:

```
4 2 1
0 0
0
0 1
1
1 0
1
1 1
0
```

Abbildung 2.21: Wertetabelle und Trainingsdatei der `XOR`-Funktion

Der vorgeschlagene Wert ist $0,7$.

Die Anfangsinitialisierung der Gewichte liegt standardmäßig zwischen $-0,1$ und $0,1$. Sie kann gegebenenfalls verändert werden.

Die Standardtransferfunktion ist die Sigmoidfunktion. In [Nis05] wird empfohlen, eine symmetrische Sigmoidfunktion mit Funktionswerten zwischen -1 und 1 statt zwischen 0 und 1 zu nutzen. Dies macht das Training schneller und genauer.

Die Anzahl der Schichten und der verdeckten Neuronen müssen experimentell ermittelt werden. Ein Ansatzpunkt kann sein, dass ein NN durch Anpassung der Gewichte lernt. Je mehr Neuronen es enthält, desto mehr Gewichte können angepasst werden, um eine Aufgabe zu lösen. Zu viele Gewichte können aber zum Problem werden, da das NN nur auf ein spezielles Szenario eingestellt wird und die Verallgemeinerung verloren geht. Es erkennt dann nur noch das Trainingsset. Dieser Effekt nennt sich Überanpassung (Overfitting). [Nis05]

2.9.7 Anwendung in der Mikrosystemtechnik

Die Anwendungen für NN sind vielfältig. Einige wurden in 1.4 bereits erwähnt. Weitere sind in [Che93] beschrieben, unter anderem die Verwendung eines NN zur Inspektion von Siliziumwafern. Das NN erkennt Fehler anhand der Tatsache, dass ein fehlerhafter Wafer anders als ein korrekter „aussieht“.

In dieser Arbeit wird ein NN zur Erkennung von Eigenmoden verwendet. Dies gleicht dem Prinzip eines Assoziativspeichers. Die Umsetzung der Eigenmodenerkennung ist in Abschnitt 4.4 beschrieben. In [Maz92] ist die Softwaresimulation eines Assoziativspeichers mittels eines NN an einem Beispiel beschrieben. Das Beispiel beinhaltet nur ein einschichtiges Netz, allerdings kann die allgemeine Vorgehensweise daran sehr gut nachvollzogen werden.

Kapitel 3

Konzept

In diesem Kapitel wird das Konzept der Optimierung von Teststrukturen und der dafür notwendigen Scritte erläutert.

Im ersten Abschnitt werden die Sollmaße einer Struktur definiert, auf deren Basis die gesamte Arbeit beruht. Danach folgt die Herangehensweise, wie Teststrukturen bewertet werden. Dieser Abschnitt ist auch ohne Optimierung, interessant, da mit der Bewertung eine Vergleichsbasis für die Strukturen geschaffen wird. In Abschnitt 3.3 schließt sich das Optimierungskonzept an. Die Abschnitte 3.4, 3.5 und 3.7 zeigen wichtige Komponenten der Optimierung konzeptuell auf. Abschließend stellt Abschnitt 3.6 Betrachtungen zur Sensitivität von Fertigungsparametern dar. Diese sind für die Umsetzung der Bewertung von Teststrukturen und damit für die Optimierung relevant.

3.1 Festlegung der Sollmaße einer Struktur

Da durch den Fertigungungprozess von MEMS Schwankungen auftreten (siehe Abschnitt 2.1), stimmen gefertigte Strukturen nicht vollständig mit dem Simulationsmodell überein.

Zur Identifizierung der Fertigungsparameter benötigt das in Abschnitt 2.5 beschriebene Verfahren eine Soll- oder Normstruktur. Als Sollmaß stehen mehrere Maße in der Entwicklung beziehungsweise Fertigung zur Auswahl.

In Abbildung 3.1 sind einige Schritte dargestellt, die Einfluss auf die Maße einer Struktur haben. Als Sollmaße einer Teststruktur werden die Abmaße des ANSYS-Modells festgelegt, da die konkreten Abweichungen während des Fertigungsprozesses nur schwer allgemeingültig erfassbar sind. Dabei summieren sich Abweichungen in der Maskenfertigung für die Lithographie und Strukturierungsverfahren durch beispielsweise DRIE auf. Die Federbreiten sind in der gefertigten Struktur im Allgemeinen geringer als im ANSYS-Modell.

Eine Simulation der Teststruktur mit Sollmaßen ergibt die Sollreihenfolge der Eigenmoden. Anhand dieser werden in den Varianzsimulationen die Eigenmoden sortiert, da es auf Grund der Parametervariation zum Vertauschen der Reihenfolge

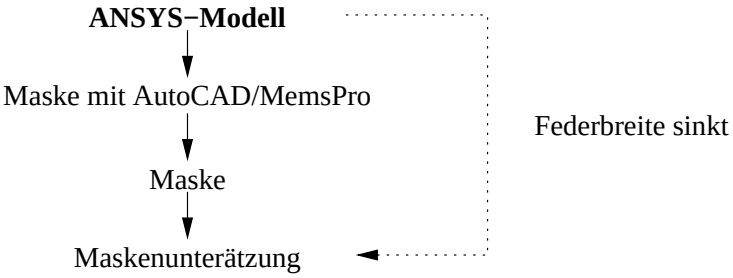


Abbildung 3.1: Abweichung vom Sollmaß am Beispiel Federbreite

der Eigenmoden kommen kann (siehe Anhang C). Aus den sortierten Eigenmoden wird das Response Surface berechnet.

3.2 Bewertung von Teststrukturen

Zum Vergleich mehrerer Teststrukturen muss jede nach verschiedenen Gesichtspunkten bewertet werden, um diese anschließend durch Wichtung untereinander einer Gesamtbewertung zu vereinen.

Ein Bewertungskriterium wird mit einem Bewerter wie in Abbildung 3.2 umgesetzt. Dieser wird durch eine Beschreibung und eine Liste aus Unterbewertern charakterisiert. Weiter besteht er aus Eingangsdaten, aus dem die Bewertungsfunktion die Ausgangsdaten berechnet. Die Eingangsdaten sind entweder beliebige Daten oder die Ausgangsdaten der Unterbewerter. In ersterem Fall kann die Liste der Unterbewerter auch leer sein.

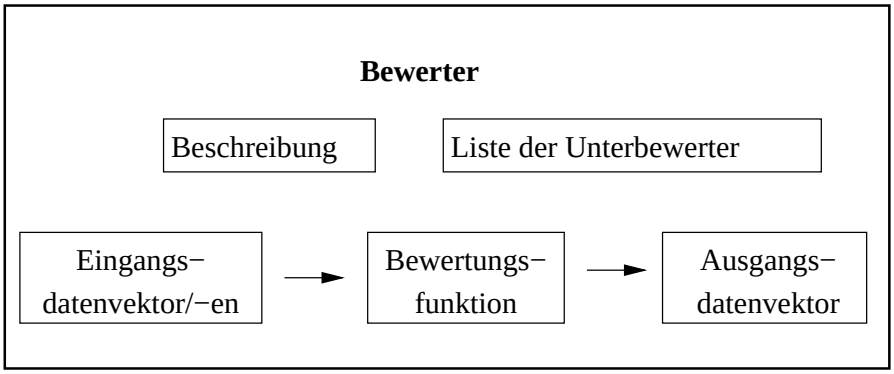


Abbildung 3.2: Elemente eines Bewerters

Die Bewertungsfunktion realisiert die Wichtung der Bewertungsparameter. Sie muss nicht linear sein oder eine stetige mathematische Funktion abbilden. Es können auch bestimmte Kombinationen von Eingangsdaten eine spezielle Bewertung erhalten.

Da jeder Bewerter als Master-Bewerter mit beliebig vielen Slave-Bewertern

(Unterbewertern) fungieren kann (siehe Abbildung 3.3), ist dieser ein Knoten eines Bewertungsbaumes. Die Slave-Bewerter werden über die Bewertungsfunktion des Master-Bewerters verknüpft. Dies wird durch alle in einen Master-Bewerter eingehenden Kanten dargestellt. Ein Bewerter ohne Slave-Bewerter bildet die Blätter eines Bewertungsbaumes.

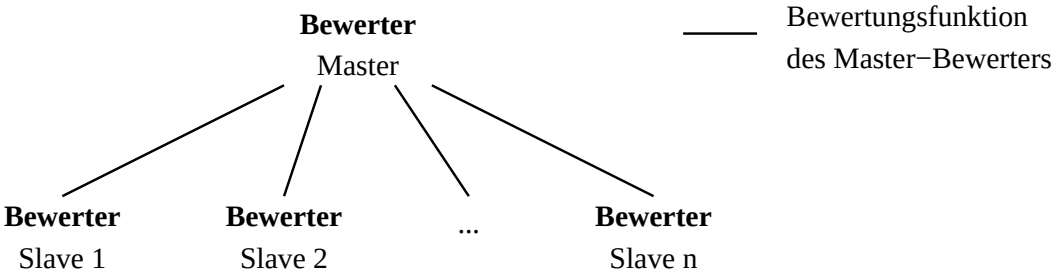


Abbildung 3.3: Teilbaum eines Bewerters

Abbildung 3.4 zeigt den Bewertungsbaum für eine Teststruktur. Basis der Bewertung bilden die Simulationsdaten mit Eigenfrequenzen und Eigenmoden. Diese bestimmen zugleich die Messbarkeit einer Teststruktur. Aus der Regression der Simulationsdaten ergeben sich die Regressionskoeffizienten und die Sensitivität der Parameter. Aus ihnen ergibt sich die Eignung für bestimmte Einsatzanforderungen. Die Wurzel des Bewertungsbaumes, das heißt die Endbewertung der Teststruktur, wird durch die *Messbarkeit* und *Eignung* festgelegt.

Die *Messbarkeit* gibt an, wie gut jede einzelne Eigenmode gemessen werden kann. Dabei gehen die Messgrenzen der Eigenfrequenzen in Verbindung mit den Eigenmoden ein, da In-Plane und Out-of-Plane nicht gleich gut gemessen werden können (siehe Abschnitt 2.3).

Die *Eignung* gibt an, in wieweit die einzelnen Eigenmode in der Lage sind, signifikante Daten für die Identifizierung eines oder mehrerer Parameter zu liefern. Dazu wird die Antwortfläche der Struktur herangezogen (siehe Abschnitt 2.5). Der Bewerter *Regressionskoeffizienten* beurteilt die Abhängigkeit und Trennbarkeit der Parameter untereinander, der Bewerter *Parametersensitivität* die Empfindlichkeit von Parametern der Struktur in den jeweiligen Eigenmoden.

Die Bewertungskriterien und ihre Wichtung müssen experimentell oder auf Grund von Erfahrung abgeschätzt werden. Eine exakte mathematische Beschreibung für den allgemeinen Fall ist nicht offensichtlich.

Jedes Individuum und damit jede Teststruktur eines Evolutionären Algorithmus benötigt eine Bewertung. Anhand dieser wird entschieden, wie gut oder schlecht das Individuum ist und wie es sich weiter entwickeln soll.

Der Vorteil dieses Verfahrens ist, dass der Bewertungsbaum einfach erweiterbar ist. Es können zusätzliche Bewertungskriterien hinzugefügt werden, ohne dass er komplett überarbeitet oder gar zerstört werden muss.

Die Bewertung erfolgt auf der Grundlage von Noten. Das Schema in Tabelle 3.1 liefert eine bessere Differentierung als “geeignet” und “ungeeignet”. Die Gesamtbe-

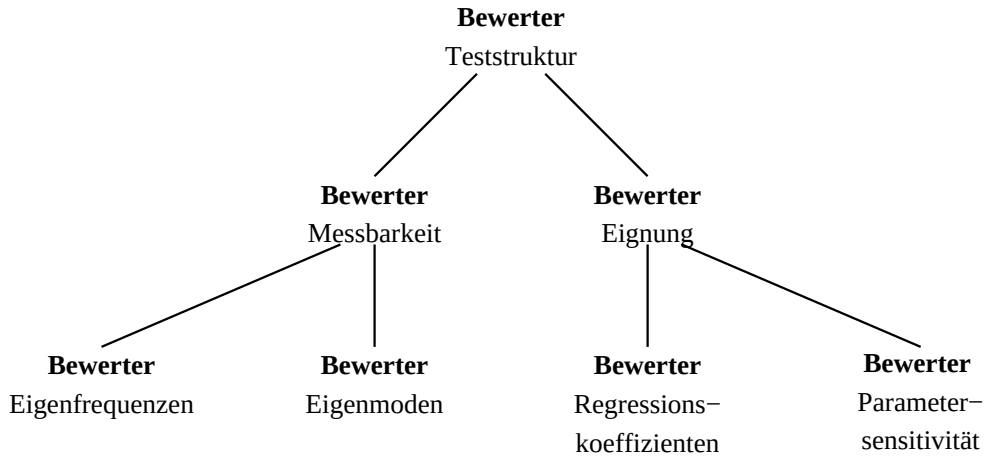


Abbildung 3.4: Bewertungsbaum für eine Teststruktur

wertung einer Teststruktur muss nicht ganzzahlig sein.

Tabelle 3.1: Notenschema für die Bewertung

Note	1	2	3	4	5
Beschreibung	sehr gut	gut	befriedigend	ausreichend	ungenügend

3.3 Optimierung von Teststrukturen

3.3.1 Ansatz

Teststrukturen sollen zur Identifizierung von Fertigungsparametern (siehe Abschnitt 2.5) genutzt werden. Deshalb werden an sie bestimmte Anforderungen gestellt. Diese gehen von den geometrischen Abmessungen einer Struktur bis zur Sensitivität gegenüber bestimmten Parametern. Eine Optimierung zielt darauf, eine geeignete Teststruktur für spezifische Randbedingungen zu finden. Die optimale Teststruktur ist nicht garantiert, jedoch wird, wie in Abbildung 3.5 dargestellt, aus einem vorhandenen beziehungsweise bestimmten Pool, die am besten geeignete ausgesucht. Diese Auswahl kann nur erfolgen, wenn die Teststrukturen vergleichbar sind. Sie müssen somit auf Grundlage der Anforderungen bewertet werden. Das Konzept der Bewertung einer Teststruktur ist in Abschnitt 3.2 beschrieben.

Für eine Optimierung wird eine große Anzahl verschiedener Teststrukturen benötigt. Diese können zum einen vom Entwickler bereitgestellt werden (vergleiche Abschnitt 2.7), zum anderen kann eine bestimmte Grundgeometrie vorgegeben und verändert werden. Wie in Abschnitt 1.5 beschrieben, können sogar neue Geometrien erzeugt werden. Zur Demonstration der prinzipiellen Vorgehensweise werden in dieser Arbeit vorhandene Grundgeometrien verändert. Abbildung 3.6 zeigt den

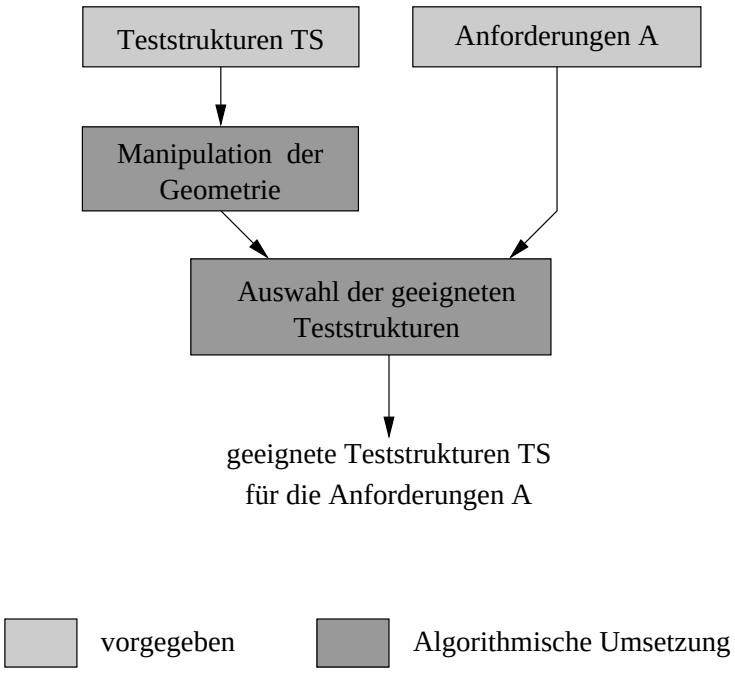


Abbildung 3.5: Auswahl einer TS für bestimmte Anforderungen

Optimierungsvorgang für Teststrukturen, welche bestimmte Anforderungen erfüllen sollen. Das Ergebnis nach mehrmaligem Durchlaufen des Algorithmus sind eine oder mehrere „gut“ geeignete Teststrukturen. Was „gut“ ist, legt die Bewertung fest.

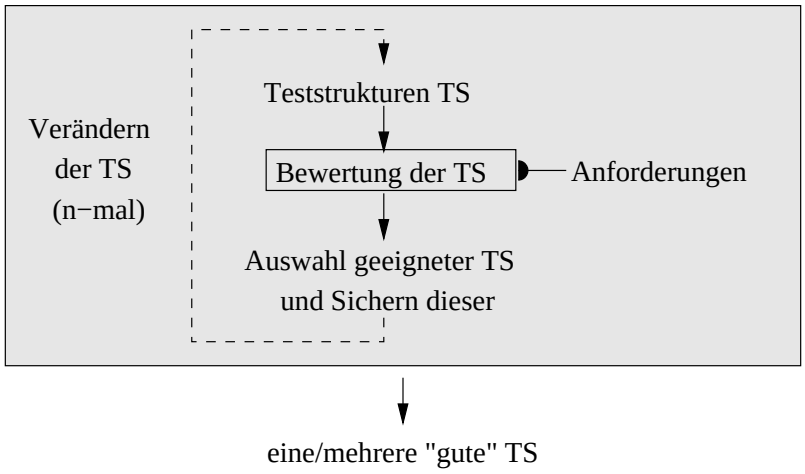


Abbildung 3.6: Optimierungsvorgang für Teststrukturen

Der Optimierungsalgorithmus kann durch einen Evolutionären Algorithmus abgebildet werden. Die Grundlagen für Evolutionäre Algorithmem sind in Abschnitt 2.8 beschrieben.

Im nächsten Abschnitt werden die Parameter einer allgemeinen Teststruktur und ihre Bedeutung bei der Optimierung erklärt. Anschließend erfolgt die Beschrei-

bung des Evolutionären Algorithmus.

3.3.2 Parameter einer Teststruktur

Die Parameter einer Teststruktur sind in Geometrie- und Fertigungsparameter unterteilbar. Die Geometrieparameter umfassen die geometrischen Abmessungen der Struktur wie laterale Ausdehnungen x_{max} und y_{max}, Abmessungen von Massen m_{x_i} und m_{y_i}, sowie Federlängen l_{x_i} und l_{y_i} (siehe Abbildung 3.7). Abstände, welche in

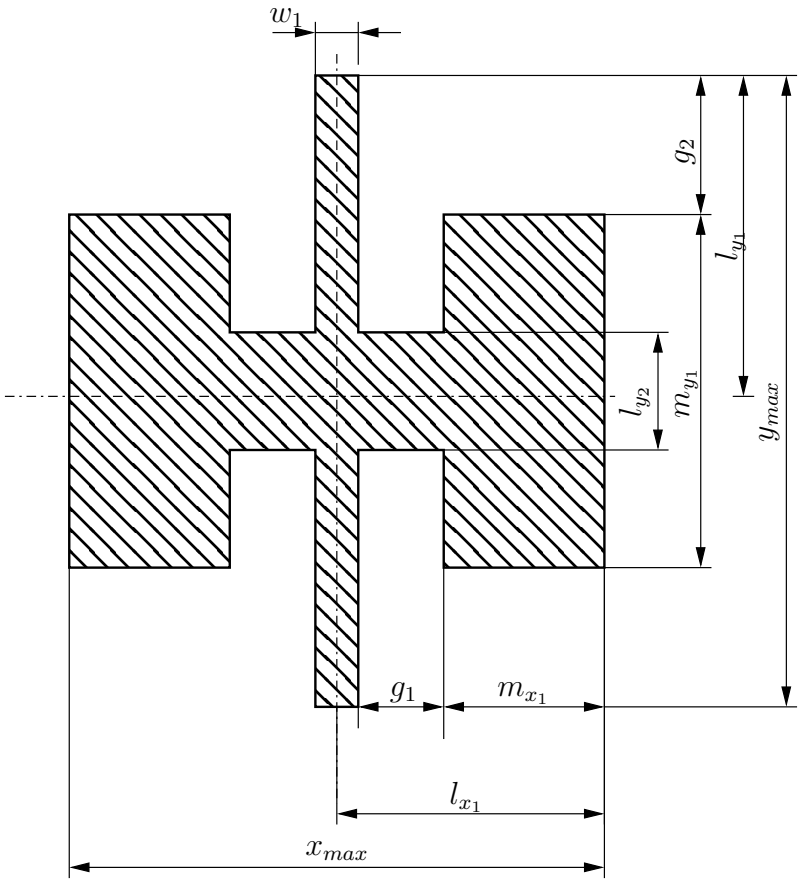


Abbildung 3.7: Geometrieparameter am Beispiel einer TS

bestimmten Intervallen liegen müssen, werden mit g_i bezeichnet. Die Federbreiten w_i dünner Federn zählen zu den Fertigungsparametern, da sie im Verhältnis zu dickeren wesentlich stärker von den Fertigungsprozessen beeinflusst werden. Das Endmaß legt der Entwickler je nach Anforderung fest. Dies könnte in diesem Fall die Endgröße der Struktur oder die Abstände g_i sein. In Abbildung 3.8 ist der Federquerschnitt einer Teststruktur dargestellt, um die Fertigungsparameter zu verdeutlichen. Die Federbreite w und die Keiligkeit x, sowie die Federdicke th werden durch Fertigungsprozesse bestimmt, welche in Abschnitt 2.1 beschrieben sind. Der Stress σ in den Federn zählt ebenso dazu.

Nach der Fertigung werden die Fertigungsparameter über eine zerstörungsfreie Parameteridentifikation ermittelt (Abschnitt 2.5). Dabei wird aus der Simulation

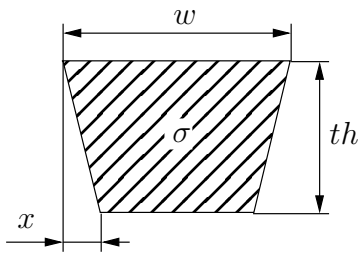


Abbildung 3.8: Fertigungsparameter am Federquerschnitt

mit verschiedenen Parameterwerten eine Antwortfläche der Teststrukturen für jede Eigenmode erzeugt.

Die Geometrieparameter werden bei der Optimierung von Teststrukturen durch den Evolutionären Algorithmus verändert.

3.3.3 Anwendung des Evolutionären Algorithmus

Die Optimierung von Teststrukturen mit einem Evolutionären Algorithmus erfolgt nach den Grundlagen aus Abschnitt 2.8.

Die Chromosomen für den Evolutionären Algorithmus werden aus allen Geometrieparametern oder nur aus einzelnen gebildet. Sollen verschiedene Teststrukturen in einem Evolutionären Algorithmus verarbeitet werden, so ist das Chromosom für jede Teststruktur gleich. In der Parametrisierung der Teststrukturen sind die Geometrieparameter entsprechend anzupassen.

Der prinzipielle Ablauf dieser Optimierung ist in Abbildung 3.9 dargestellt.

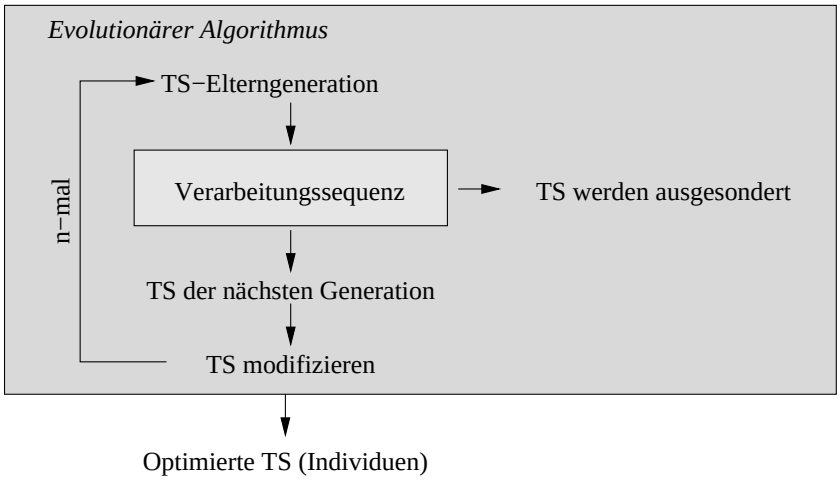


Abbildung 3.9: Optimierung von TS mittels EA

Der Evolutionäre Algorithmus läuft für n Generationen. Das Ergebnis sind für die Anforderungen geeignete Teststrukturen. Die konkrete Ausprägung des Evoluti-

onären Algorithmus, das heißt ob und wie viele Individuen sterben und auf welche Weise die nächste Generation gebildet wird, ist vom verwendeten Evolutionären Algorithmus abhängig (siehe Abschnitt 2.8).

Die in Abbildung 3.10 dargestellte Verarbeitungssequenz zeigt die für jedes Individuum ablaufenden Schritte, welche zur Bewertung führen. Diese Bewertung und die Prüfung der Gene auf Gültigkeit beeinflussen den Ablauf des Evolutionären Algorithmus.

Die Genprüfung ist notwendig, um zulässige Geometrien zu erhalten. Die Randbedingungen müssen zuvor festgelegt werden. In der Umsetzung in Abschnitt 4.2.3 wird näher darauf eingegangen. Die Bewertung einer Teststruktur wird in Abschnitt

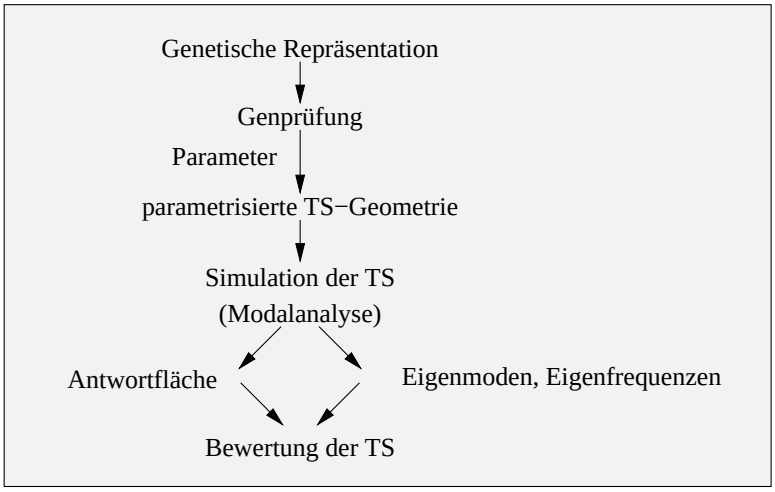


Abbildung 3.10: Verarbeitungssequenz einer TS

3.2 erläutert.

In Abschnitt 4.2 ist eine konkrete Umsetzung für das beschriebene Verfahren aufgezeigt. Prinzipiell liegen ihm, wie zuvor beschrieben, Teststrukturen zugrunde, deren Geometrieparameter über einen Evolutionären Algorithmus verändert werden. In diesem Algorithmus findet für jede Teststruktur eine Bewertung statt. Diese ergibt sich aus spezifischen Anforderungen wie zum Beispiel Messbarkeit der Teststruktur und Eignung für ein bestimmtes Problem.

Der Evolutionäre Algorithmus liefert nach einer festzulegenden Anzahl an Generationen eine Reihe von Teststrukturen, welche für die Anforderungen geeignet sind. Diese Teststrukturen müssen, wie erwähnt, nicht zwangsläufig optimal sein (siehe Abschnitt 2.8). Das verwendete Modell eines Evolutionären Algorithmus in dieser Arbeit ist der Genetische Algorithmus.

3.4 Parametervariation in ANSYS

Für das in Abschnitt 2.5 beschriebene Verfahren zur Parameteridentifikation wird die theoretische Antwort eines Mikrosystems benötigt. Grundlage dessen ist die

in ANSYS simulierte Struktur. Für diese wird ein Eigenmode und die zugehörige Eigenfrequenz für mehrere verschiedene Werte der Fertigungparameter (siehe Abschnitt 3.3.2) berechnet. Aus diesen Daten wird durch Regression die Antwortfläche ermittelt (siehe Anhang A).

In diesem Abschnitt soll das Konzept der Parametervariation in ANSYS am Beispiel von drei Fertigungsparametern erläutert werden. Jeder dieser Parmeter p wird zwischen dem Minimalwert $p_{\min}$ und dem Maximalwert $p_{\max}$ um die Schrittweite p_{step} verändert.

Das Modell der zu simulierenden Struktur ist parametrisiert. Die Parameter für einen Schritt werden, wie in Abbildung 3.11 dargestellt, als Argumente übergeben. Anschließend wird das Modell mit der entsprechenden Strukturdicke th und der

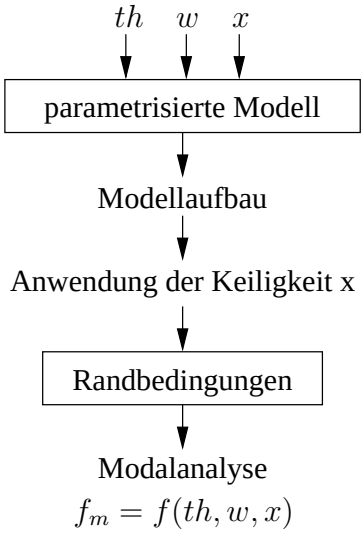


Abbildung 3.11: Einzelner Schritt der Parametervariation

Federbreite w aufgebaut. Die Keiligkeit x wird in einem separaten Schritt, entsprechend Abschnitt 3.5.2, umgesetzt. Danach erfolgt das Anlegen der Randbedingungen und die Modalanalyse für mehrere Eigenmoden. Der Ergebnisvektor $\vec{r}$ beinhaltet die Parametersets und den Vektor der Eigenfrequenzen $\vec{f}$.

$$\vec{r} = (th, w, x, f_1, f_2, \cdots, f_i)$$

Diese Strukturanalyse wird, wie in Listing 3.1 dargestellt, für verschiedene Parametersets durchgeführt. Die Gesamtanzahl der Simulationen n ergibt sich aus

$$\begin{aligned} n &= \prod n_{p_i} = n_{th} \cdot n_w \cdot n_x \\ \text{mit } n_{p_i} &= ((p_{i_{\max}} - p_{i_{\min}} DIV p_{i_{\text{step}}}) + 1 \end{aligned} \tag{3.1}$$

mit p_i ... Parameter i
n_{p_i} ... Anzahl der Variationen der Parameters i
DIV ... ganzzahlige Division mit Rest.

Listing 3.1: Parametervariation in Pseudokode

```
th = th_min
w = w_min
x = x_min

wiederhole für th solange th ≤ th_max
        wiederhole für w solange w ≤ w_max
                wiederhole für x solange x ≤ x_max

                        führe Strukturanalyse des Modells mit th, w, x
                            durch
                        th = th + th_step
                        w = w + w_step
                        x = x + x_step

                wiederhole_ende
        wiederhole_ende
wiederhole_ende
```

Die Mindestanzahl K der Simulationen wird durch die Regressionsfunktion bestimmt (siehe Abschnitt A.1). Für drei Parameter und eine lineare Regressionsfunktion wären theoretisch acht simulierte Parametersets ausreichend.

Die Variation der Parameter hat bei Strukturen mit hohem Aspektverhältnis Einfluss auf das Verhalten. Dabei kann ein Tausch der Eigenmoden auftreten. Für die Regression ist jedoch eine gleichbleibende Reihenfolge notwendig, daher müssen die Eigenmoden vorher gegebenenfalls sortiert werden. Sortiergrundlage bildet die *Normstruktur* mit dem *Normparameterset*. Dieses wird zuvor festgelegt. In Abschnitt 4.4 wird näher darauf eingegangen. Um die Wahl der Regressionsfunktion nicht von vorn herein einzuschränken, ist es oft sinnvoll, mehr Parametersets als theoretisch notwendig zu simulieren.

Die Umsetzung der Parametervariation in ANSYS ist in Abschnitt 4.2.5 beschrieben.

3.5 Realisierung des Parameters Keiligkeit

Wie in Abschnitt 2.1 beschrieben, verursacht das verwendete Fertigungsverfahren eine Maskenunterätzung und eine Keiligkeit. Dies beeinflusst das Verhalten von Strukturen mit hohem Aspektverhältnis in ihrem Verhalten.

Die in Abschnitt 3.1 festgelegten Sollmaße legen die betrachteten Maskenmaße fest. Die Keiligkeit wird in der Simulation berücksichtigt, da sie das Strukturverhalten maßgeblich beeinflusst.

3.5.1 Möglichkeiten der Realisierung

Die Einarbeitung in den Modellaufbau einer Struktur würde diesen verkomplizieren. Die unabhängige Berechnung der Keiligkeit k ist daher günstiger. Sie ist abhängig von der Strukturdicke th. Ihr Betrag x ist über eine vorzugebende Funktion $k(th)$ berechenbar. Die Richtung ist gesondert zu ermitteln.

In der Simulation sollen nur jene senkrechten Flächen betrachtet werden, welche die Struktur vom umgebenden Medium trennen (siehe Abbildung 3.12). Weiterhin

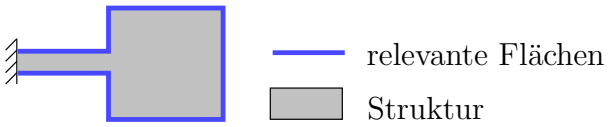


Abbildung 3.12: Berücksichtigte Flächen für die simulierte Keiligkeit

wird die Unterätzung, welche die Keiligkeit verursacht, zur Vereinfachung als isotrop und konstant angenommen.

In [Str07] ist ein Verfahren für die Umsetzung in der Simulation beschrieben. Dieses setzt jedoch die vorherige Selektion der betroffenen Flächen vorraus, was in einem automatisierten Ablauf nicht möglich ist, da diese sich ändern können. Außerdem steigt der Zeitaufwand des Verfahrens mit steigender Strukturvernetzung, da die Keiligkeit durch Verschiebung von Knoten der außenliegenden Elemente erfolgt.

Ein anderer Ansatz ist die nachfolgend erläuterte thermische Analogie. Dabei wird der Temperaturgradient zur Richtungsbestimmung der Keiligkeit genutzt. Für eine konkrete Struktur muss die Richtungsbestimmung nur einmal erfolgen, da die Grundstruktur bei einer Parametervariation (siehe 3.4) gleich bleibt. Der Vorteil dieses Verfahrens liegt in der Strukturunabhängigkeit, es funktioniert zuverlässig und stellt keine Restriktionen an den Modellaufbau. Weiterhin erfolgt der Ablauf automatisch, was für den Einsatz in einem automatisierten Optimierungsverfahren entscheidend ist.

Das Konzept der thermischen Analogie wird im folgenden Abschnitt erklärt, die Umsetzung in ANSYS findet sich in Abschnitt 4.3.

3.5.2 Konzept der Thermischen Analogie

Basis für die Richtungsermittlung der Keiligkeit ist die Wärmeleitung in einem Medium. Die für die Wärmeleitung maßgebende Stoffeinheit ist die Wärmeleitfähigkeit λ. Generell lässt sich die Temperatur in einem Körper als orts– und zeitabhängige Funktion $T(x, y, z, t)$ darstellen. Der transiente Fall soll hier allerdings nicht von Interesse sein. Wenn die Temperatur örtlich nicht gleich verteilt ist, strömt Wärme vom Bereich höherer Temperatur zum Bereich tieferer Temperatur. Dieser Wärmestrom Q ist proportional zum Gradienten der Temperatur $-\text{grad}\ T$. Das Vorzeichen resultiert aus der Definition, dass der Gradient in Richtung der ansteigenden Temperatur gerichtet ist. Der Proportionalitätsfaktor ist die Wärmeleitfähigkeit λ. Sie

ist im Allgemeinen von der Temperatur abhängig, wird jedoch in vielen technischen Berechnungen als konstant angenommen. Der stationäre Fall kann mit Gleichung 3.2 beschrieben werden.

$$\nabla(\lambda \nabla T) = f(x, y, z) \tag{3.2}$$

mit λ ... Wärmeleitfähigkeit
T ... Temperatur
∇ ... Nabla-Operator.

Ist die Wärmeleitfähigkeit im gesamten Berechnungsgebiet konstant, so kann die vereinfachte Gleichung 3.3 verwendet werden.

$$\lambda \nabla^2 T = f(x, y, z) \tag{3.3}$$

Nach der Umwandlung mit der Methode der finiten Elemente in ein Gleichungssystem, welches in [GM98] beschrieben wird, ergibt sich für den stationären Fall in Matrizenschreibweise

$$[k_{\text{therm}}] \cdot \{T\} = -\{Q\}$$

mit $[k_{\text{therm}}]$ ... Matrix der Wärmeleitfähigkeiten
$\{T\}$ ... Vektor der Temperaturen
$\{Q\}$ ... Vektor der von außen zu– oder abgeführten Wärmeströme.

Der Vektor $\{Q\}$ ergibt sich aus den Wärmestromwerten aller Knoten eines Modelles. [GM98]

Durch die Simulation, welche in [GM98] näher beschrieben ist, können bei gegebenen Temperaturen und Wärmeleitfähigkeiten die zu den Knoten gehörenden Temperaturgradienten ermittelt werden. Die Richtung des Temperaturgradienten wird der Richtung der Keiligkeit zugeordnet (siehe Abbildung 3.13). Dafür wird die

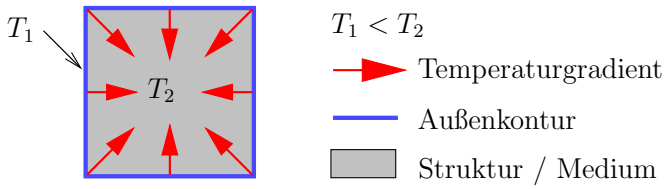


Abbildung 3.13: Temperaturgradienten in einer Struktur

Außenkontur einer Struktur auf die Umgebungstemperatur gesetzt, die Struktur auf eine höhere Temperatur. Der Temperaturgradient zeigt somit immer vom Rand in die Struktur.

Es werden acht Richtungen betrachtet. Diese werden durch die Komponenten t_x und t_y des Temperaturgradienten entsprechend Tabelle 3.5.2 bestimmt. Zur Identifizierung dienen Richtungskodes.

In der Simulation wird für jeden Punkt auf der Außenkontur der Richtungskode ermittelt. Anhand dessen werden die Punkte durch die Unterätzung nach Tabelle 3.5.2 um Δu_x in x-Richtung und Δu_y in y-Richtung verschoben.

In Abschnitt 4.3 ist die Umsetzung in ANSYS beschrieben.

Tabelle 3.2: Ermittlung der Richtungskodes

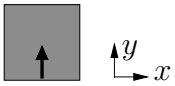
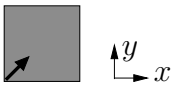
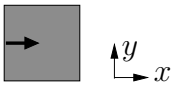
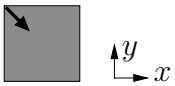
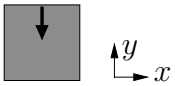
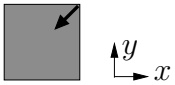
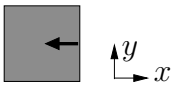
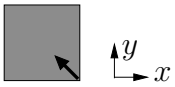
t_x	t_y	Richtung	Richtungskode
$= 0$	> 0		1
> 0	> 0		2
> 0	$= 0$		3
> 0	< 0		4
$= 0$	< 0		5
< 0	< 0		6
< 0	$= 0$		7
< 0	> 0		8

Tabelle 3.3: Verschiebung anhand von Richtungskodes

Richtungskode	Δu_x	Δu_y
1	0	x
2	x	x
3	x	0
4	x	$-x$
5	0	$-x$
6	$-x$	$-x$
7	$-x$	0
8	$-x$	x

3.6 Sensitivitätsbetrachtungen

Die Sensitivitäten der Teststrukturen nach einem Parameter ergeben sich aus der Differentiation der Antwortfläche nach dem jeweiligen Parameter. Prinzipiell gilt, dass eine höhere Sensitivität besser ist, als eine niedrigere. Im folgenden wird angenommen, dass die Frequenzänderung für kleine Parameteränderungen linear.

Abbildung 3.14 zeigt die Frequenzabhängigkeit eines Parameters p. Es soll eine Parametervariation detektiert werden, die eine Frequenzabweichung Δf von der Normeigenfrequenz f_{norm} verursacht.

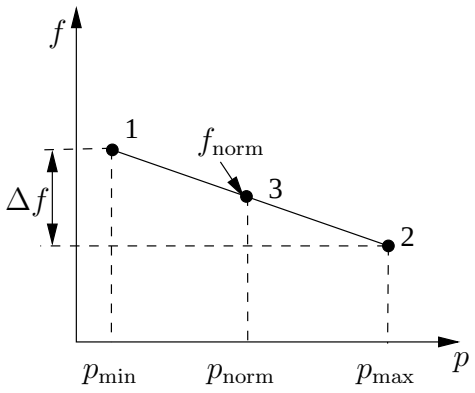


Abbildung 3.14: Frequenzabweichung bezüglich der Normfrequenz

Für die minimale Frequenzänderung Δf über dem gesamten Parametervariationsbereich ergeben sich Einschränkungen aus der Simulation, der Messung und der Auflösung des Parameters (siehe Gleichung 3.4). Im Folgenden werden diese näher erläutert.

$$\Delta f \geq max(\Delta f_{\text{sim}}, \Delta f_{\text{mess}}, \Delta f_{\text{res}}) \tag{3.4}$$

mit Δf_{sim} ... minimal notwendige Frequenzänderung aus Simulation
Δf_{mess} ... minimal notwendige Frequenzänderung aus Messung
Δf_{res} ... minimal notwendige Frequenzänderung aus Parameterauflösung

Frequenzabweichungen aus der Simulation können durch Simulations- und Regressionsfehler eingebracht werden. Die Abweichung durch die Parametervariation muss deutlich größer sein als diese Fehler, damit sie tatsächlich durch die Variation der Parameter und nicht durch andere Störfaktoren verursacht wird. Die minimale relative Frequenzabweichung Δf_{sim} wird auf 0,1% der Normfrequenz (siehe Gleichung 3.5) festgelegt. Damit liegt sie erfahrungsgemäß über den durch die erwähnten Störfaktoren erzeugten Frequenzschwankungen.

$$\begin{aligned} \Delta f &\geq \Delta f_{\text{sim}} \\ \Delta f_{\text{sim}} &= 0,1\% \cdot f_{\text{norm}} \end{aligned} \tag{3.5}$$

Die Eigenfrequenzen bei Extremwerten der Parameter müssen sich weiterhin deutlich von der Normeigenfrequenz abheben, um in der Messung mit dem LDV erfasst werden zu können. Abbildung 3.15 zeigt diese absolute Frequenzverschiebungen anhand des Amplitudengangs.

Mit der in Tabelle 4.1 getroffenen Annahme symmetrischer Parameterabweichungen gilt im Folgenden Gleichung 3.6. Damit ergibt sich die durch die Messung bedingte minimale Frequenzabweichung Δf_{mess} (siehe Gleichung 3.7).

$$\Delta f_{\text{abs}} = \Delta f_{\text{abs,1}} = \Delta f_{\text{abs,2}} \tag{3.6}$$

$$\Delta f_{\text{mess}} = 2 \cdot \Delta f_{\text{abs}} \tag{3.7}$$

Wird die der Amplitudengang manuell ausgewertet, liegt die minimale erkennbare Frequenzdifferenz höher, als bei der Auswertung mittels Lorentzian Fit (siehe

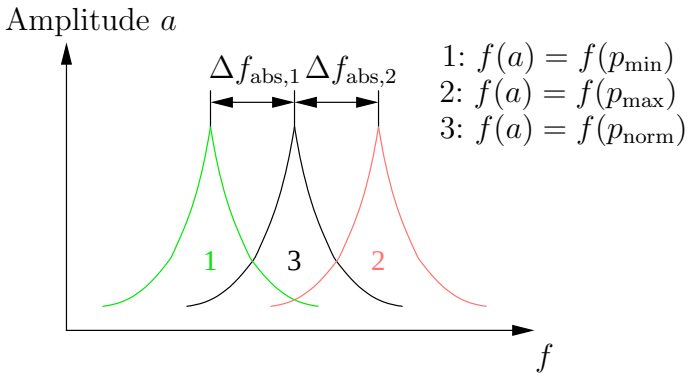


Abbildung 3.15: Absolute Frequenzverschiebung zur Normfrequenz

Tabelle 3.4). Ausführungen über die Messung von Resonanzfrequenzen und Lorentzian Fit finden sich in [LTC+03] und [PA98].

Tabelle 3.4: Auswerteverfahren und Grenzen von Δf_{abs}

Auswerteverfahren	Bereich von Δf_{abs}
Manuell	$20\,Hz \ldots 1\,kHz$
Lorentzian	$1\,Hz \ldots 20\,Hz$

Je höher die Frequenzabweichung Δf_{abs}, desto leichter kann die Variation des Parameters ausgewertet werden. Der Parameter mit einer bestimmten Auflösung Δp bestimmt die Frequenzabweichung Δf_{res}. Diese muss im messbaren Bereich liegen (siehe Tabelle 3.4). Ist der Variationsbereich des Parameters als seine Auflösung, so ist dies das beschränkende Kriterium.

Die tatsächliche Frequenzabweichung Δf durch die Sensitivität des Parameters und damit durch den Anstieg m der Antwortfläche gegeben. Bei linearer Abhängigkeit des Parameters von der Frequenz kann mit den Differenzenquotienten gearbeitet werden. Es gilt Gleichung 3.8. Da $p_{\max}$ und $p_{\min}$ feststehen, ist Δf durch Gleichung 3.9 definiert. Für die Betrachtung der Parameterauflösung erfolgt für $x = \Delta p$ analog.

$$\begin{aligned} m &= \frac{\Delta y}{\Delta x} \\ \text{mit } \Delta y &= \Delta f \\ \Delta x &= p_{\max} - p_{\min} \end{aligned} \tag{3.8}$$

$$\Delta f = m \cdot (p_{\max} - p_{\min}) \tag{3.9}$$

Eine Struktur kann danach bewertet werden, ob ihre Eigenfrequenzabweichung bei Variation der Parameter Gleichung 3.4 einhält und wie groß die Abweichung zur minimal notwendigen Frequenzdifferenz ist. In dieser Arbeit wird eine Parameterauflösung von $50\,nm$ angenommen.

3.7 Eigenmodenerkennung

Wie in Abschnitt 2.5 erläutert, kann es bei der Parametervariation zu einem Vertauschen der Eigenmoden kommen. Geschieht dies, steigt der Regressionsfehler der Antwortfläche.

Die Eigenmodenerkennung soll vertauschte Moden erkennen und korrigieren. Dies kann, wie im folgenden Abschnitt beschrieben, manuell erfolgen. Für die Automatisierung muss eine andere Lösung angestrebt werden. Eine Möglichkeit ist die in Abschnitt 3.7.2 dargestellte Erkennung mit Hilfe eines Neuronalen Netzes.

Die Erkennung ist auch in der Messtechnik interessant, um Eigenmoden eindeutig identifieren zu können. Abschnitt 3.7.2 liefert dazu einen Ansatz.

3.7.1 Manuelle Eigenmodenerkennung

Die manuelle Eigenmodenerkennung erfolgt durch die Betrachtung signifikanter Punkte der Struktur bei jeder Eigenfrequenz. Die normierten Amplituden der Auslenkung dieser werden durch die Normstruktur festgelegt. Es wird ein Bedingungsbaum erstellt, der jede Eigenmode anhand der Auslenkung in diesen Punkten identifiziert.

Die Eigenmoden jeder Struktur mit von den Normparametern abweichenden Parametern werden über diesen Baum geprüft. Vertauschte Eigenmoden können somit erkannt und korrigiert werden.

Der Nachteil dieser Methode ist, dass der Bedingungsbaum für jede Normstruktur neu zu erstellen ist. Bei komplexen Strukturen steigt auch seine Komplexität an und ist in manchen Fällen nicht durch wenige Punkte charakterisierbar.

Diese Methode ist somit für die automatisierte Optimierung nicht geeignet.

3.7.2 Eigenmodenerkennung mittels Neuronalem Netz

Die Eigenmodenerkennung mit einem Neuronalen Netz kann zum einen zur Sortierung von Eigenmoden einer simulierten Struktur anhand einer Normstruktur verwendet werden. Zum anderen ermöglicht sie die Identifizierung gemessener Eigenmoden. Die Identifizierung erfolgt auf Basis der normierten Auslenkung bei einer Eigenfrequenz. Diese Auslenkung ist bei kleinen Geometrieabweichungen für jeden Eigenmode gleichartig. Gleichartig bedeutet in diesem Fall, dass die Auslenkungen zwar betragsmäßig unterschiedlich sein und wertemäßig geringfügig abweichen können, aber von der Gesamttendenz gleichbleiben. Die Modenerkennung von simulierten Eigenmoden aus der Parametervariation ist in Abbildung 3.16 dargestellt. Grundlage ist dabei die zu betrachtende Teststruktur. Für diese wird ein Normparameterset, wie in Abschnitt 3.3.2 eingeführt, festgelegt. Die Struktur mit diesem Parameterset bestimmt die Reihenfolge der Eigenmoden. Die Reihenfolge der Eigenmoden aus der Parametervariation muss dieser für die spätere Berechnung der Antwortfläche, angepasst werden. Eine abweichende Reihenfolge macht sich durch

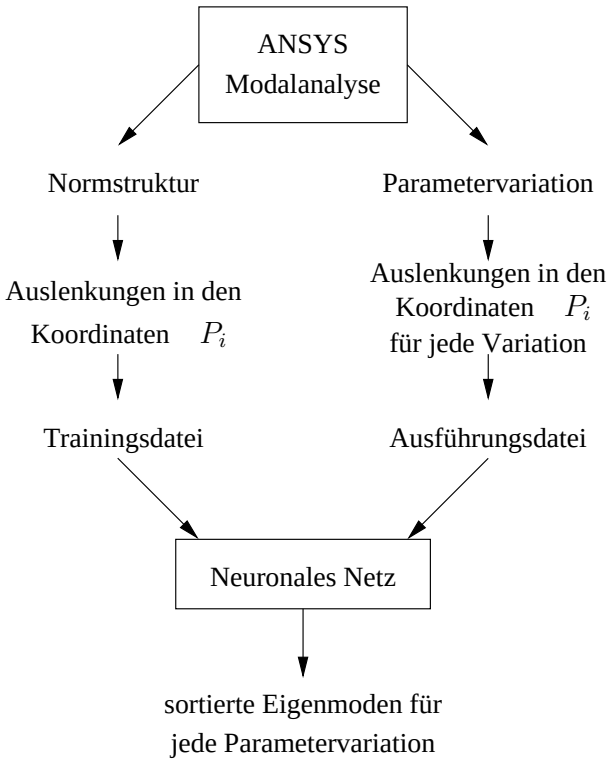


Abbildung 3.16: Ablauf Sortierung der Eigenmoden der Parametervariation

eine hohe Abweichung von der Antwortfläche bemerkbar.

Mit den Daten aus dem Normparameterset wird das Neuronale Netz trainiert, beim Ausführen mit den Daten der Parametervariation werden deren Moden entsprechend erkannt. Die Daten bestehen aus den jeweiligen Auslenkungen an den Koordinaten P_i. Die Struktur der Daten wird in Abschnitt 4.4.2 näher erläutert. Das Ergebnis ist eine Liste der sortierten Eigenmoden für die jeweilige Parametervariation.

Die Eigenmodenerkennung von gemessenen Strukturen erfolgt wie in Abbildung 3.17 dargestellt. Die Bereitstellung der Trainingsdaten erfolgt analog zur Modenerkennung von simulierten Eigenmoden. Die Koordinaten der verwendeten Auslenkungen müssen mit denen der Messung korrespondieren. Die Daten aus der Messung müssen aufbereitet werden. Um Vergleichbarkeit zu gewährleisten, werden die Auslenkungen normiert und für das Neuronale Netz entsprechend Abschnitt 4.4.2 formatiert. Das mit den Daten der simulierten Normstruktur trainierte Neuronale Netz liefert beim Ausführen dieser Daten die erkannten Eigenmoden.

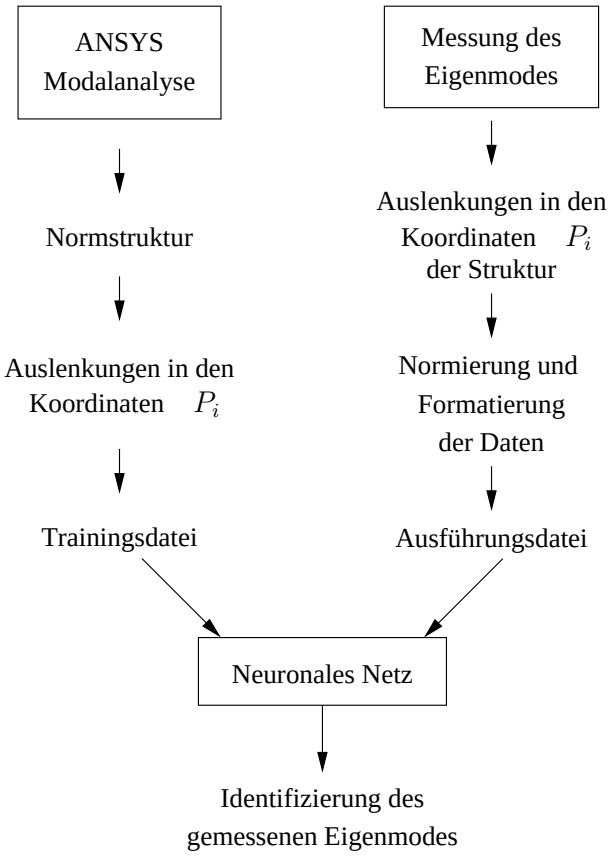


Abbildung 3.17: Ablauf Identifizierung der gemessenen Eigenmoden

Kapitel 4

Umsetzung

In diesem Kapitel wird die Umsetzung der in Kapitel 3 beschriebenen Konzepte erläutert.

Im ersten Abschnitt wird eine mögliche Bewertung von Teststrukturen dargestellt. Diese Bewertung ist eigenständig und kann auch ohne Genetischen Algorithmus genutzt werden. Der zweite Abschnitt behandelt die Umsetzung der Optimierung anhand eines Testframeworks. Dieses zielt darauf, die Bewertung in der Optimierung von Teststrukturen zusammen mit einem Genetischen Algorithmus zu verwenden. Auf die dabei verwendeten einzelnen Komponenten wird kurz eingegangen.

Die darauffolgenden Abschnitte zeigen die detaillierte Umsetzung zur Lösung wichtiger Aufgabenstellungen bei der Entwicklung und Optimierung von Teststrukturen. Die Parametervariation in ANSYS (Abschnitt 4.2.5) und die Umsetzung der Keiligkeit (Abschnitt 4.3), sowie die Eigenmodenerkennung (Abschnitt 4.4) sind auch ohne Verwendung des Optimierungsframeworks von Bedeutung, weshalb sie, die Parametervariation ausgenommen, getrennt erläutert werden.

Im letzten Abschnitt erfolgt eine Auflistung der in der Arbeit betrachteten Teststrukturgeometrien.

Tabelle 4.1 zeigt die für diese Arbeit angenommenen Schwankungen der Fertigungsparameter. Sie sind die Grundlage für die Parametervariation und die Bewertung der Sensitivitäten der Parameter.

Tabelle 4.1: Angenommene Fertigungsparameterschwankungen

Parameter	Minimalwert	Normwert	Maximalwert	Absolute Schwankung
Federbreite w / μm	4,5	5	5,5	1
Strukturdicke th / μm	48	50	52	4
Keiligkeit x / μm	0,1	0,2	0,3	0,2

4.1 Bewertung von Teststrukturen

Die Bewertung einer Teststruktur erfolgt hierarchisch, wie im Konzept in Abschnitt 3.2 erläutert. In den folgenden Abschnitten wird die Umsetzung dieser Hierarchie sowie die Verbindung der einzelnen Ebenen ausgehend von der Gesamtbewertung einer Teststruktur dargestellt.

Die Umsetzung der Bewertung orientiert sich an der in Abschnitt 2.1 vorgestellten Fertigungstechnologie und den in Tabelle 4.1 angenommenen Schwankungen der Fertigungsparameter. Eine Beispielbewertung einer Teststruktur findet sich in Anhang F.

4.1.1 Gesamtbewertung einer Teststruktur

Die Gesamtbewertung einer Teststruktur ergibt sich aus den Bewertungen ihrer Eigenmoden. Jede betrachtete Eigenmode wird in Bezug auf Messbarkeit und Eignung entsprechend der beiden folgenden Abschnitte beurteilt.

Gleichung 4.1 zeigt die Bewertung b einer Eigenmode i. Sie wird auf ganze Zahlen gerundet.

$$b_i = \frac{M_i + E_i}{2} \tag{4.1}$$

mit b_i ... Bewertung der i-ten Eigenmode
M_i ... Bewertung der Messbarkeit der i-ten Eigenmode
E_i ... Bewertung der Eignung der i-ten Eigenmode

Die Gesamtbewertung B berechnet sich nach Gleichung 4.2 aus dem Mittelwert eines Teils der Eigenmodenbewertungen b. Die Noten des Eigenmodenbewertung werden aufsteigend sortiert. Für die Parameteridentifikation wird eine minimale Anzahl geeigneter Eigenmoden $m_{\min}$ benötigt. Für die Teststruktur wird der Durchschnitt aus den $m_{\min}$-besten Eigenmoden gebildet. Gibt es weniger als $m_{\min}$ Eigenmoden, die eine Noten besser als 5 haben, wird die gesamte Struktur mit ungenügend bewertet. Es wird ein benötiger Eigenmode für jeden zu identifizierenden Parameter angesetzt. Damit entspricht m_min der Anzahl der variierten Fertigungsparameter.

$$B = \begin{cases} 5 & \text{, wenn} \quad m_{\text{gut}} < m_{\min} \\ \frac{\sum_{i=1}^{m_{\min}} bs_i}{m_{\min}} & \text{sonst} \end{cases} \tag{4.2}$$

mit B ... Gesamtbewertung der Teststruktur
bs_i ... sortierte Bewertung der i-ten Eigenmode
m ... Anzahl der betrachteten Eigenmoden
m_{gut} ... Anzahl „guter“Eigenmoden mit $b_i \leq 4$
$m_{\min}$ ... Mindestanzahl Eigenmoden für die Parameteridentifikation

Die Gesamtbewertung wird für eine besseren Differenzierung auf eine Dezimalstelle gerundet.

4.1.2 Bewertung der Messbarkeit

Die Bewertung der Eigenfrequenzen, der Hauptbewegungsrichtung der Eigenmoden und der Messbarkeit orientiert sich an Tabelle 2.2. Die Bewertung der Frequenzen F, dargestellt in Tabelle 4.2, und der Hauptbewegungsrichtung der Eigenmoden H, siehe Tabelle 4.3, gehen zusammen in die Messbarkeit ein.

Das Notenspektrum kann für sie nicht vollständig ausgenutzt werden, da nicht genügend Intervalle zur Verfügung stehen. Eigenfrequenzen f_i außerhalb des Messbereichs erhalten die Note 5, sie sind nicht nutzbar.
Eigenfrequenzen im Bereich zwischen $100\,kHz$ und $500\,kHz$ bekommen die Note 4, da in diesem nur Out-of-Plane-Eigenmoden zuverlässig gemessen werden können. Die verbleibenden beiden Bereiche werden entsprechend besser bewertet. Von den Hauptbewegungsrichtungen der Eigenmoden wird Out-of-Plane besser gestellt. Sie kann in mehreren Frequenzbereichen erfasst werden.

Tabelle 4.2: Bewertung Frequenz

Frequenzbereich / kHz			Note
$0 <$	f_i	≤ 10	1
$10 <$	f_i	≤ 100	2
$100 <$	f_i	≤ 500	4
	f_i	> 500	5

Tabelle 4.3: Bewertung Eigenmoden

Bewegungsform	Note
Out-of-Plane	1
In-Plane	2

Die Bewertung der Messbarkeit M ergibt sich nach Gleichung 4.3. Ein Eigenmode i wäre genau dann nicht messbar, wenn die zugehörige Eigenfrequenz f_i außerhalb des Messbereichs liegt oder wenn es sich um einen In-Plane Mode handelt, der im nichtmessbaren Bereich liegt. Ist der Eigenmode messbar, so wird seine Messbarkeit aus dem Durchschnitt der beiden oben genannten Unterbewertungen. Die Bewertung wird auf ganze Zahlen gerundet.

$$M_i = \begin{cases} 5 & \text{, wenn} \quad F_i = 5 \vee H_i = 5 \\ 5 & \text{, wenn} \quad F_i = 4 \wedge H_i = 2 \\ (F_i + E_i)/2 & \text{sonst} \end{cases} \tag{4.3}$$

mit i ... i-ter Eigenmode
M ... Bewertung Messbarkeit
F ... Bewertung Frequenz
H ... Bewertung Bewegungsform des Eigenmodes

4.1.3 Bewertung der Eignung

Die Bewertung der Eignung E einer Teststruktur drückt aus, in wieweit sich ein bestimmter Parameter mit ihr identifizieren lässt. Dafür muss eine bestimmte Sensitivität bezüglich diesem gegeben und die Wechselwirkung zwischen ihm und anderen

Parametern gering sein. Die Betrachtungen erfolgen anhand der Antwortfläche der Teststrukturen für jeden Eigenmode.

Idealerweise ist eine Struktur bei einem Eigenmoden von genau einem Parameter abhängig. In der Realität ist dies selten der Fall. Aus diesem Grund soll die Bewertung der Eignung aus dem maximalen Notenabstand der Sensitivitäten jeweils zweier Parametern ermittelt werden (siehe Gleichung 4.4). Der Notenabstand ergibt sich nach Gleichung 4.5. Er wird für jede Kombination zweier Parameter berechnet.

$$E_i = max(e_{i,1}, \ldots, e_{i,v}) \tag{4.4}$$

$$e_{i,j} = 5 - |S_{p,i,a} - S_{p,i,b}| \; \forall \, a, b \in 1, \ldots, n \wedge a \neq b \tag{4.5}$$

mit i ... i-ter Eigenmode
E_i ... Bewertung der Eignung
$e_{i,j}$ ... Bewertung Notenabstand
n ... Anzahl der Parameter
$S_{p,i,a}$ ... Sensitivität der Parameters p_a bei einem Eigenmode i
$S_{p,i,b}$ ... Sensitivität der Parameters p_b bei einem Eigenmode i
v ... Anzahl der Parameterpaarkombinationen

Die Umsetzung im Optimierungsframework ist auf zwei Parameter beschränkt.

Die Ermittlung der benötigten Sensitivität für die Parameteridentifikation wird in Abschnitt 3.6 erläutert. Darauf aufbauend ergibt sich die Bewertung S_p der Sensitivität eines Parameters p nach Gleichung 4.6 aus der Gesamtfrequenzschwankung $\Delta f_{\mathrm{ges,i}}$ (Gleichung 4.7) und der Frequenzschwankung über die Parameterauflösung (Gleichung 4.8). Die Intervalle gehen aus den Auswerteverfahren für die Eigenfrequenzen und deren Einschränkungen hervor (siehe Tabelle 3.4).

$$S_{\mathrm{p,i}} = \begin{cases} 1 & \text{, wenn} \quad \Delta f_{\mathrm{res,i}} > 1\,kHz \\ 2 & \text{, wenn} \quad 20\,Hz < \Delta f_{\mathrm{res,i}} \leq 1\,kHz \\ 3 & \text{, wenn} \quad 1\,Hz < \Delta f_{\mathrm{res,i}} \leq 20\,Hz \\ 5 & \text{, wenn} \quad \Delta f_{\mathrm{res,i}} \leq 1\,Hz \\ 5 & \text{, wenn} \quad \Delta f_{\mathrm{ges,i}} < 0,001 \cdot f_{\mathrm{norm}} \end{cases} \tag{4.6}$$

mit i ... i-ter Eigenmode
$\Delta f_{\mathrm{ges,i}}$ ... Frequenzschwankung über die Parametervariationsbreite
$\Delta f_{\mathrm{res,i}}$ ... Frequenzschwankung über die Parameterauflösung
$f_{\mathrm{norm,i}}$ ... Normeigenfrequenz
m_i ... Anstieg der Antwortfläche für den Parameter p

$$\Delta f_{\mathrm{ges,i}} = |m_i \cdot \Delta p| \tag{4.7}$$

$$\Delta f_{\mathrm{res,i}} = |m_i \cdot \Delta p_{\mathrm{res}}| \tag{4.8}$$

mit Δp ... Parametervariationsbreite $p_{\max} - p_{\min}$
Δp_{res} ... Parameterauflösung

Für kleine Wechselwirkungen zwischen Parametern müssen die Kreuzterme der Regressionsfunktion klein sein. Kreuzterme sind diejenigen Regressionsglieder,

in denen mehr als ein Parameter einen Exponenten größer oder gleich eins aufweist. Gleichung 4.9 zeigt eine Beispielbewertung K der Kreuzterme.

$$
\begin{aligned}
K_i &= \frac{\sum_{j=1}^{N_k} k_j}{N_k} \\
k &= \begin{cases} 1 & \text{, wenn} \quad \kappa = 0 \\ 2 & \text{, wenn} \quad 0 < \kappa \leq 0.125 \cdot a_{\min} \\ 3 & \text{, wenn} \quad 0.125 \cdot a_{\min} < \kappa \leq 0.25 \cdot a_{\min} \\ 4 & \text{, wenn} \quad 0.25 \cdot a_{\min} < \kappa \leq 0.5 \cdot a_{\min} \\ 5 & \text{, wenn} \quad 0.5 \cdot a_{\min} < \kappa \geq a_{\min} \end{cases} \\
a_{\min} &= min(a_{p,1}, \ldots, a_{p,n})
\end{aligned}
\tag{4.9}
$$

mit i ... i-ter Eigenmode
K ... Bewertung der Kreuzterme
N_k ... Anzahl der Kreuzterme
k ... Einzelbewertung Kreuzterm
κ ... Regressionskoeffizient des Kreuzterms
a_p ... Regressionskoeffizient des Parameters p
n ... Anzahl der Parameter

Die Kreuzterm werden in der Umsetzung zu Vereinfachung nicht berücksichtigt.

4.2 Optimierung von Teststrukturen

Die Optimierung von Teststrukturen erfolgt nach dem in Abschnitt 3.3 aufgezeigten Konzept. Für die konkrete Umsetzung sind mehrere Bausteine notwendig. Im folgenden Abschnitt werden die verwendeten Programme und Programmiersprachen für die jeweiligen Bausteine erläutert.

4.2.1 Komponenten der Optimierung

Die einzelnen Komponenten der Optimierung bedürfen zum Teil unterschiedlicher Programme, da kein Programm beziehungsweise keine Beschreibungssprache existiert, die alle notwendigen Lösungsansätze bereitstellt. Sie müssen weiterhin performant sein und Schnittstellen untereinander ermöglichen. Die entsprechende Aufgliederung ist in Abbildung 4.1 dargestellt. Für die Simulation der Teststrukturen ist zwingend ein entsprechendes Programm notwendig, in diesem Fall ANSYS. Es bietet den Vorteil, auch ohne graphische Nutzeroberfläche (Graphical User Interface (GUI)) bedienbar zu sein. Dieser Modus wird im folgenden als Batch-Modus bezeichnet. Die Steuerung erfolgt über sogenannte Makros. Diese ASCII-Dateien beinhalten Steuerkommandos und können im Batch-Modus, gegebenenfalls mit Argumenten, geladen werden. Die Geometrien der Teststrukturen werden ebenfalls in Makros bereitgestellt.

ANSYS wird über eine Skriptsprache gesteuert, die Abarbeitung erfolgt sequenziell. Weiterhin steht ein Interpreter der Skriptsprache Tcl [WJ03] zur Verfügung, mit der auch graphische Oberflächen realisiert werden können.

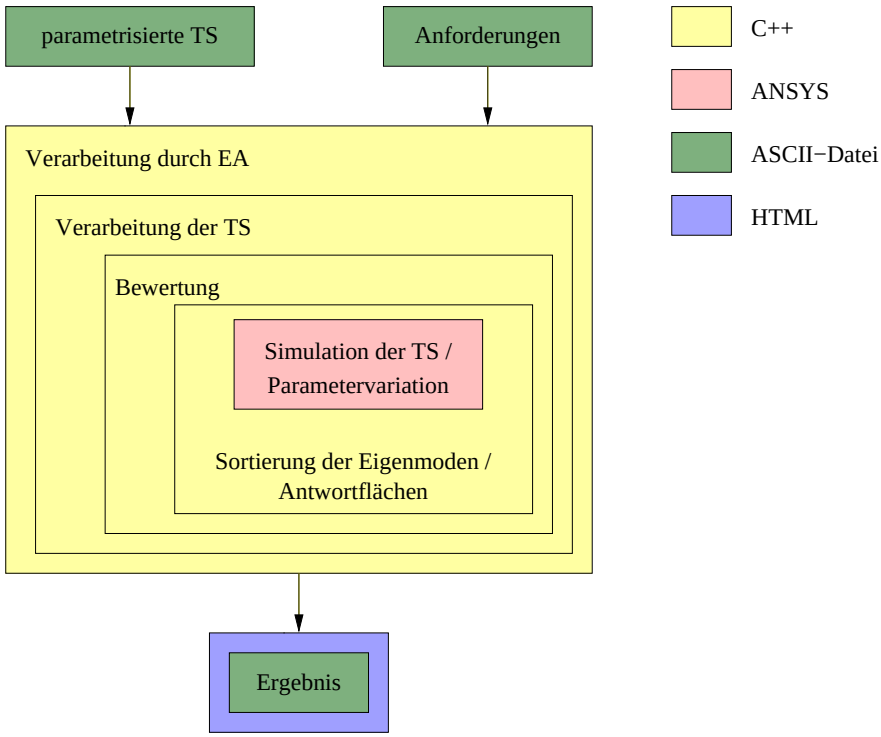


Abbildung 4.1: Bausteine der Optimierung einer TS

Obwohl eine einheitliche Implementierung in einer Sprache wünschenswert ist, wurde in dieser Arbeit davon abgesehen. Zum einen ist die Ausführung von Skriptsprachen gegenüber vorübersetzen Sprachen im Allgemeinen langsamer. Weiterhin ist ein objektorierentierter Ansatz in ANSYS beziehungsweise Tcl aufwendig zu realisieren. Zudem gibt es keine gut dokumentierten und leistungsfähigen Bausteine für Neuronale Netze und Evolutionäre Algorithmen in Tcl, welche die Anforderungen erfüllen. Die Implementierung dieser wäre zwar prinzipiell möglich, würde aber den Rahmen dieser Arbeit sprengen und könnte nicht mit der Performance und der Stabilität bereits entwickelter und getesteter Bibliotheken konkurrieren. Aus diesen Gründen wurden der EA und die Modenerkennung mittels Neuronalem Netz in der Programmiersprache C++ [Str03] umgesetzt. Die für den EA verwendetete Bibliothek ist in Abschnitt 2.8.5 erklärt, die für das Neuronale Netz verwendete in Abschnitt 2.9.6.

C++ wurde wegen der zahlreich verfügbaren Bibliotheken und des Sprachumfangs gewählt. Theoretisch wären auch andere Sprachen und Programme wie MATLAB [ÜKP05] verwendbar. Diese müssen allerdings die folgenden Komponenten übsetzen können und eine Schnittstelle zu ANSYS ermöglichen. Zusätzlich muss die Verarbeitung einer Generation des EA performant erfolgen, da der Zeitbedarf für die gesamte Optimierung mit der Anzahl der Generationen steigt. Der Anstieg kann nicht näher spezifiert werden, da die Verarbeitungszeit einzelner Komponenten, z. B. der Eigenmodenerkennung mittels Neuronalem Netz, nicht konstant ist (vergleiche Abschnitt 5.1.1).

Da der die Optimierung umschließende EA in C++ umgesetzt ist, liegt es

nahe, für die restlichen Komponenten ebenfalls diese Sprache zu nutzen und damit Schnittstellenprobleme weitestgehend zu vermeiden. Dies hat den Vorteil, dass eine einfach Steuerung aller in C++ möglich ist.

Die Plattformunabhängigkeit ist gewährleistet, in sofern keine plattformspezifischen Befehle verwendet werden. Gegebenenfalls müssten vorhandene Codepassagen angepasst und der Quellcode neu übersetzt werden.

Die Komponente „Bewertung“, welche vom EA benötigt wird, führt die Bewertung einer TS nach dem in Abschnitt 3.2 vorgestellten Konzept durch. Grundlage dafür sind die Antwortfläche und die Eigemoden einer TS. Die Daten dafür werden in Form von Dateien durch das aufgerufene ANSYS erzeugt, welches wie erwähnt über Makros gesteuert wird. Die Antwortfläche wird mit C++ erzeugt, da die Eigenmoden aus der Parametervariation zuerst mit dem Neuronalen Netz sortiert werden. Dies könnte auch in ANSYS geschehen, allerdings müsste es dazu erneut aufgerufen werden. Der verfolgte Ansatz ist zeit- und ressourcensparender.

Der Datenaustausch erfolgt stehts über ASCII-Dateien. Dies hat den Vorteil, dass die Dateien ohne Zusatzsoftware plattformunabhängig lesbar sind. Zudem erleichtert es die Fehlersuche.

Die Präsentation der Ergebnisse erfolgt über ASCII-Dateien und wird zusätzlich über HTML aufbereitet. Mit HTML können sie besser gruppiert und übersichtlich dargestellt werden. Es lässt sich auf jeder Plattform mit einem entsprechendem Browser darstellen. Es wird von diesem interpretiert und benötigt somit keine weitere Behandlung.

4.2.2 Das Optimierungsframework

Das eingesetzte Buildsystem ist das plattformunabhängige, Opensource-Programm Cross-plattform Make (CMake). Es generiert aus einer plattform- und kompilerunabhängigen Projektbeschreibung plattformabhängige Eingabedateien für Entwicklungsumgebungen. In diesen Projektdateien müssen keine weiteren Einstellungen vorgenommen werden. Die Projektbeschreibung erfolgt mit Konfigurationsdateien (sogenannte `CMakeLists.txt`-Dateien). Es werden unter anderem die Umgebungen make, Visual Studio, Eclipse und Xcode unterstützt. Die eingängige Syntax ermöglicht eine einfache Erstellung modularer Programmstrukturen (Bibliotheken und ausführbare Dateien) und Installationspaketen. Bibliotheksabhängigkeiten im System und Includepfade können mit den mitgelieferten Tests geprüft werden, weiterhin werden Unittests unterstützt, welche Testprogramme für den Programmcode ausführen und das Ergebnis zusammenfassen. Eigene Skripte sind einbaubar, und ebenfalls plattformunabhängig. Eine umfangreiche Dokumentation sowie zahlreiche Beispiele sind verfügbar. Dieses Buildsystem wird von einem der größten Opensourceprojekte der Welt, dem KDE-Projekt, genutzt.[MH07]

Für mathematische Berechnungen, speziell Matrizenrechnung, wird die C++-Bibliothek `NEWMAT` [NEW] genutzt. Operationen auf dem Dateisystem werden mit der C++-Bibliothek `boost_filesystem` [Boo] umgesetzt. Dies ermöglich eine einfachere Portierung auf ein anderes Betriebssystem. Auf diese Bibliotheken wird nicht

näher eingegangen, für Einzelheiten sei auf die jeweilige Handbücher verwiesen. Die Entwicklung erfolgte in dieser Arbeit unter Linux.

Wie bereits im vorherigen Abschnitt erwähnt, ist das Optimierungsframework objektorierentiert in C++ umgesetzt. Der Aufbau ist modular, so dass einzelne Komponenten zum Teil eigenständig nutzbar beziehungsweise vollständig austauschbar sind. Auf einzelne Klassen wird an dieser Stelle nicht eingegangen, dafür sei auf die Dokumentation [Opt] verwiesen.

Das Framework besteht neben den vorgestellten Bausteinen aus weiteren Hilfskomponenten. Diese ermöglichen beispielsweise Funktionen wie das Verwenden von Konfigurationsdateien und Statusausgaben. Die Unterteilung erfolgt in die Optimierungskomponenten und die Hilfskomponenten Konfiguration und HTML-Ausgabe.

Die Optimierungskomponenten sind hierarchisch aufgebaut (siehe Abbildung 4.2). Die Wurzel bildet der Genetische Algorithmus, welcher mehrere Teststruk-

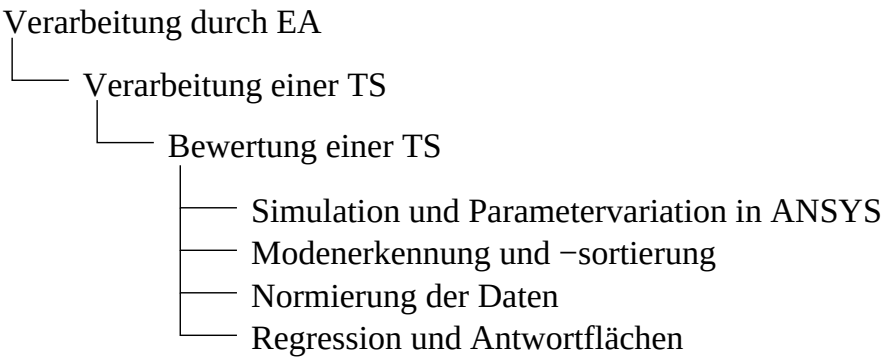


Abbildung 4.2: Hierarchie der Optimierungskomponenten

turen verarbeitet. Die Verarbeitung einer TS wird durch ihre Bewertung geprägt. Die Regression ist der Bewertung untergeordnet, ebenso wie die dafür nötige Datenerzeugung durch die Simulation mit ANSYS und die Datenaufbereitung. Für eine Optimierung ohne GA müssen die Daten, welche aus dem Genom stammen, andersweitig bereitgestellt werden.

In den folgenden Abschnitten werden die einzelnen Komponenten von der Wurzel ausgehend näher erläutert.

4.2.3 Komponente „Genetischer Algorithmus“

Für den EA wird die C++-Bibliothek GAlib (siehe Abschnitt 2.8.5) verwendet. Sie setzt die für den Evolutionären Algorithmus notwendige Funktionalität um.

Da keine neuen Teststrukturgeometrien erzeugt werden sollen, sind vollparametrisierte Grundgeometrien vorzugeben. Das Benennungsschema sollte sich an Abschnitt 3.3.2 orientieren. Die in dieser Arbeit beispielhaft untersuchten Teststrukturen sind viertelsymmetrisch. Die Gemeinsamkeit gleichzeitig untersuchter

Strukturen ist ihr Genom. Dieser besteht aus ausgewählten Parametern der Grundgeometrie. Die bereitzustellenden Daten sind:

1. ein vollparametrisiertes Modellmakro für ANSYS
2. die Randbedingungen für die Modalanalyse in ANSYS
3. eine Datei mit Genotyp (Parameternamen, leerzeichengetrennt)
4. Gültigkeitsbedingungen für den Genotyp in C++
5. eine Konfigurationsdatei

Die Umsetzung der Optimierung mit einem GA erfolgt nach dem in Abschnitt 3.3.3 beschriebenen Konzept. Der vom GA erzeugte Genotyp definiert das parametrisierte Modell für die ANSYS-Simulation. Zuvor erfolgt die Abprüfung auf die Zulässigkeit der Parameter. Nur so kann gewährleistet werden, dass korrekte Strukturen entstehen. Diese Genprüfung ist strukturabhängig und kann somit nicht verallgemeinert werden. Ihre Umsetzung erfolgt direkt in C++.

Die Bewertung einer TS wird durch die Komponente „Verarbeitung einer TS“ ermittelt. Die Ergebnisse werden in HTML zusammengestellt.

Anhang G zeigt die Verwendung des GA am Beispiel einer Demonstrationsstruktur. Er verdeutlicht die Optimierung einer Teststruktur gegen vorgegebene Zielwerte von Größe und erster Eigenfrequenz. Dieses Demonstrationsbeispiel benutzt nicht die Komponente „Verarbeitung einer TS“, sondern eine einfachere Bewertung.

Im Optimierungsframework ist die Klasse `Perfom_GA` zuständig für die Umsetzung und Ausführung des Genetischen Algorithmus.

4.2.4 Komponente „Verarbeitung einer Teststruktur“

In der Verarbeitung einer TS werden die untergeordneten Komponenten abgearbeitet. Abbildung 4.3 zeigt den abstrahierten Ablauf. Mit den Daten aus der Simulation in ANSYS wird die Erkennung der Eigenmoden durchgeführt. Daraus ergibt sich ihre Sortierung. Nach der Normierung der Daten erfolgt die Erzeugung der Antwortfläche und daraufhin die Bewertung. Die Ergebnisse werden am Ende in HTML zusammengefasst.

Abbildung 4.4 zeigt eine ausführlichere Version der Verarbeitung. Das parametrisierte Teststrukturmodell wird mit den Geometrieparametern aus dem Genom des EA in ANSYS simuliert. Das Verfahren zur Abbildung der Keiligkeit ist in Abschnitt 3.5 erläutert. Es wird nur einmal für jedes Individuum durchgeführt, um die Richtung der Unterätzung zu bestimmen. Anschließend folgt die Simulation der Normstruktur, welche die Standardreihenfolge der Eigenmoden festlegt.

Die Parametervariation ist in Abschnitt 4.2.5 näher erläutert, die anschließende Sortierung der Eigenmoden mittels NN in Abschnitt 4.2.6.

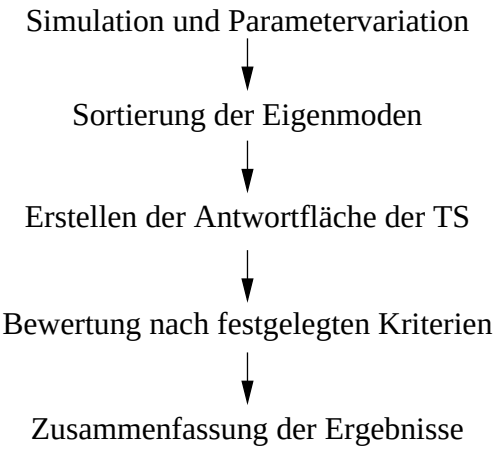


Abbildung 4.3: Ablauf der Verarbeitung einer TS (abstrahiert)

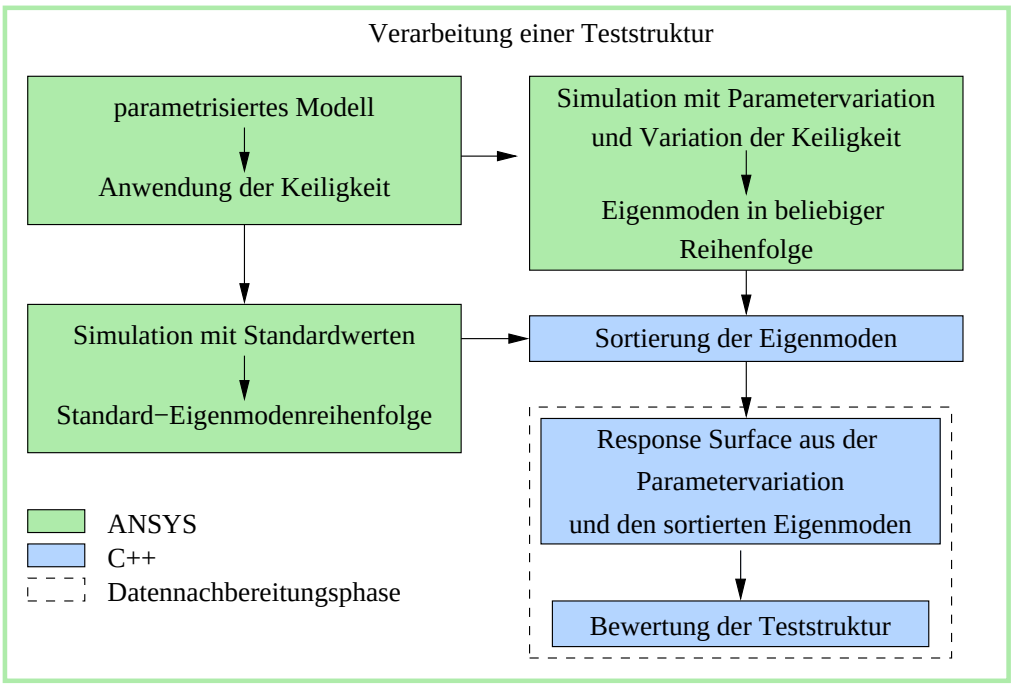


Abbildung 4.4: Verarbeitungssequenz für eine TS

Aus den sortierten Eigenmoden wird die Antwortfläche der TS für einzelne Eigenmoden nach dem Verfahren in Anhang A erzeugt. Auf Grundlage dieser Daten erfolgt die Bewertung der TS.

Die Schrittfolge für die Datennachbereitungsphase einer TS nach der Simulation ist in Abbildung 4.5 dargestellt. Daraus wird deutlich, das der Regression eine Normierung der Daten zugrunde liegt. Diese wird bereits in der Verarbeitung festgesetzt, weshalb die zugehörige Komponente ihr untergeordnet ist. Die Bewertung erfolgt abschließend nach den in Abschnitt 4.1 beschriebenen Kriterien. Die Datenaufbereitung erfordert keine Berechnungen mehr.

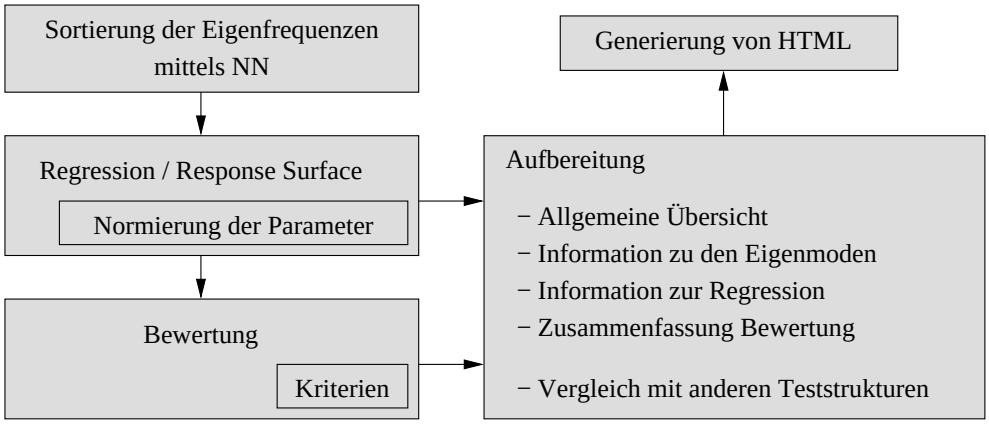


Abbildung 4.5: Datennachbearbeitungsphase einer TS

Wird diese Komponente eigenständig genutzt, so werden folgende Dateien benötigt:

1. ein parametrisiertes Modellmakro der TS für ANSYS
2. Makros mit Randbedingungen der TS für ANSYS
3. eine Konfigurationsdatei der TS

Das Modellmakro muss nur hinsichtlich der zu betrachtenden Fertigungsparameter variabel gestaltet sein.

`Process_TS` ist die für die Verarbeitung von Teststrukturen verantwortliche Klasse. Die Sortierung der Eigenmoden erfolgt durch `Train_NN` und `Exec_NN`, die Regression durch `Regression` und die Normierung durch `Normalization`. Die Klasse `Evaluation` beinhaltet das Grundgerüst für eine Bewertung, die konkrete Bewertung wird von ihr abgeleitet und angepasst. Die Zusammenfassung in HTML erfolgt über die Klasse `Generate_Html_Summary`. Nähere Erläuterungen finden sich in den Abschnitten über die entsprechenden Komponenten.

4.2.5 Komponente „Parametervariation“

Die Umsetzung der in Abschnitt 3.4 vorgestellten Parametervariation erfolgt mit Hilfe von Makros. Diese enthalten Anweisungen für ANSYS in ANSYS Parametric Design Language (APDL).[ANS]

Der Aufbau ist modular. Damit ist das Gerüst für beliebige, entsprechend parametrisierte Strukturen nutzbar. Abbildung 4.6 zeigt die Verknüpfung der Makros untereinander, Tabelle 4.4 gibt die Argumente und eine kurze Beschreibung für jedes Makro an.

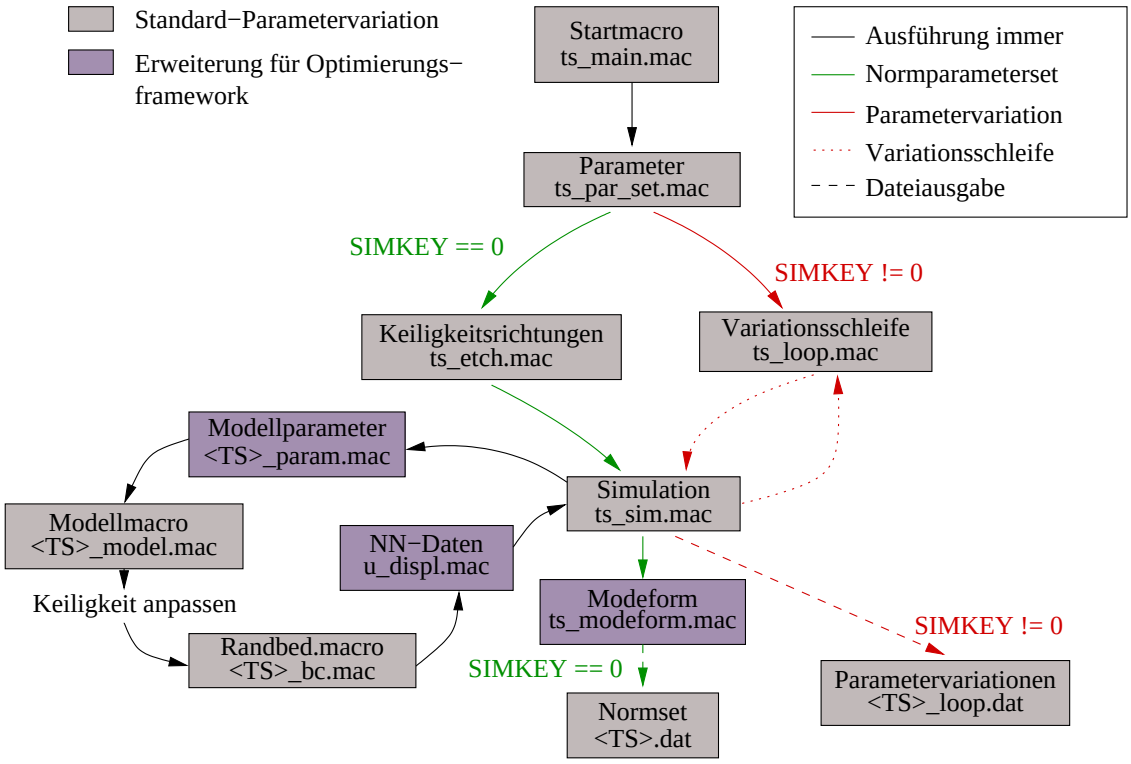


Abbildung 4.6: Abhängigkeiten der ANSYS-Makros für die Parametervariation

Mit dem Hauptmakro `ts_main.mac` startet die Parametervariation. Es bekommt als Argument den Namen der Teststruktur TS und legt für sie einen neuen Ordner an. In diesem wird auch die Simulation für die TS durchgeführt. Das Modellmakro mit der parametrisierten Geometrie der Struktur ist mit `<TS>_model.mac` und das Makro mit dem Randbedingungen mit `<TS>_bc.mac` zu benennen. Das Makro `<TS>_param.mac` enthält die Geometrieparameter des Modells. Für <TS> ist der Name der Teststruktur einzusetzen.

In `ts_par_set.mac` werden die Parameter mit Norm- und Variationswerten, ebenso wie die zu simulierende Anzahl Eigenmoden, festgelegt. Der Schalter `SIMKEY` dient zur Auswahl der Simulation der Normparametersets (`SIMKEY` = 0) beziehungsweise der Parametervariation (`SIMKEY` != 0). Vor einer Parametervariation muss stets zuerst eine Simulation der Normstruktur erfolgen, da über diese die Verschiebungsrichtungen für die Keiligkeit durch `ts_etch.mac` bestimmt werden. Das

Tabelle 4.4: Beschreibung und Argumente der Makros

Makro	Argumente	Beschreibung
<TS>_model.mac	-	Modelldatei der TS
<TS>_bc.mac	-	Datei mit Randbedingungen (boundary conditions)
<TS>.dat	-	Speicherdatei des Parametersets und der Eigenfrequenzen der Normstruktur
<TS>_loop.dat	-	Speicherdatei der Parametersets und der Eigenfrequenzen der Variationsstrukturen
<TS>_param.mac	-	Geometrieparameter für des Modell
ts_etch.mac	TS	Ermitteln der Verschiebungsrichtungen für die Keiligkeit
ts_loop.mac	TS	Durchführen der Parametervariation
ts_main.mac	TS	Hauptmakro (Laden der Parameter, Simulation des Normparametersets oder der Parametervariation)
ts_modeform.mac	-	Ermittlung der Hauptbewegungsrichtung der Eigenmoden
ts_par_set.mac	TS	Parameterdatei
ts_sim.mac	TS,$p_1, \ldots, p_n$	Geometrie laden und Modalanalyse
u_displ.mac	-	Ermittlung der Knotenverschiebungen in den Eigenmoden für das NN

Verfahren ist in Abschnitt 4.3 aufgezeigt.

Das Makro `ts_sim.mac` führt die eigentliche Simulation durch. Es realisiert den Modellaufbau durch <TS>`_model.mac`, setzt die Keiligkeit für den aktuellen Wert x um und legt die Randbedingungen mittels <TS>`_bc.mac` fest. Danach erfolgt die Modalanalyse mit den aktuellen Fertigungsparametern $p_1, \ldots, p_n$. Die Ergebnisdaten werden in <TS>`.dat` für das Normparameterset beziehungsweise in <TS>`_loop.dat` für die Variationen gespeichert.

Die Variation der Parameter wird über `ts_loop.mac` gesteuert. Dieses Makro enthält die Schleifen, welche in Listing 3.1 im Abschnitt 3.4 in Pseudokode beschrieben sind.

Die für die Optimierung verwendete Parametervariation in ANSYS ist eine Erweiterung dieses Verfahrens. In der Simulation jedes Parameterssets werden die Auslenkungen der Struktur bei jedem Eigenmode mit `u_displ.mac` erfasst. Das Normparameterset bildet damit die Normauslenkungen bei jeder Eigenfrequenz, mit welcher die der restlichen Parametersets verglichen werden. Der Vergleich beziehungsweise die Eigenmodenerkennung erfolgt mit Hilfe eines Neuronalen Netzes (siehe Abschnitt 4.4).

Das Makro `u_displ.mac` liest die Auslenkungen von festgelegten Knoten bei jeder Eigenfrequenz und speichert sie in einem für das NN geeigneten Format ab.

Für jedes Parameterset wird somit eine Datei mit den Auslenkungen und der dazugehörigen Eigenfrequenz erfasst. Abschnitt 4.4.2 befasst sich mit der konkreten Formatierung und der Auswahl der betrachteten Knoten für die Auslenkungen.

Die Geometrieparameterdaten werden durch den GA erzeugt und im Makro <TS>_param.mac abgespeichert. Sie werden in der ANSYS-Simulation geladen. Die Simulation wird durch einen Aufruf aus C++ heraus gestartet.

4.2.6 Komponente „Modenerkennung mittels NN“

Die Komponente „Modenerkennung mittels NN“ realisiert die automatisierte Erkennung und Sortierung von Eigenmoden einer Struktur. Dies erfolgt nach dem in Abschnitt 3.7 beschriebenen Konzept. Allgemeine Betrachtungen zur Umsetzung, sowie der Aufbau von Ein- und Ausgangsdaten des NN finden sich in Abschnitt 4.4.

Die objektorientierte Umsetzung findet sich in den Klassen `Train_NN`, zum Trainieren des NN mit den Normstrukturdaten, und `Exec_NN`, zum Ausführen des NN mit dem Daten aus der Parametervariation. Die Dateinamen der Ausführungsdaten können in einer Datei abgelegt werden. Die Abarbeitung erfolgt dann automatisch. Die Klasse `Exec_NN` liefert eine Datei mit allen verarbeiteten Ausführungsdaten und den entsprechend sortierten Eigenfrequenzen, sowie eine Datei, in welche nicht korrekt erkannte Parametersets entfernt wurden. Letztere wird für die Regression der Antwortfläche genutzt. Eine Logdatei protokolliert den Ablauf der Modenerkennung, sowie die Ausgaben jedes Neuronalen Netzes und die daraus resultierende Eigenmodenzuordnung.

4.2.7 Komponente „Regression und Antwortflächen“

Die Antwortflächen einer Struktur repräsentieren die Eigenfrequenzen in Abhängigkeit von Fertigungsparametern. Sie werden durch eine Regressionsanalyse (siehe Abschnitt 2.4) gewonnen. Die Ausgangsdaten liefert die Parametervariation (siehe Abschnitt 4.2.5).

Die Regression erfolgt über die Klasse `Regression`. Normierungen können mit der Klasse `Normalization` realisiert werden. Das Regressionspolynom wird mit Hilfe der Vandermond'schen Matrix ermittelt (siehe Anhang A).

Die Regressionsklasse berechnet zuerst auf Basis der gegebenen Parametersets und der zugehörigen Eigenfrequenzen die Exponentenmatrix der Vandermond'schen Matrix nach dem in Anhang B beschriebenen Verfahren. Anschließend werden die Regressionskoeffizienten ermittelt und die Daten normiert. Die Berechnung des Regressionsfehlers ist optional.

Für lineare Regressionsfunktionen können über eine Funktion die Sensitivitäten der Parameter ermittelt werden. Sie ergeben sich aus der Ableitung der Regressionsfunktion nach einem Parameter. Für höherwertige Regressionen ist die Ableitung nicht mehr trivial. Untersuchungen des Regressionsfehlers in Abhängigkeit des Regressionsgrades haben gezeigt, dass bei den betrachteten Teststrukturen eine lineare

Regression ausreichend ist (siehe Abbildung 4.7).

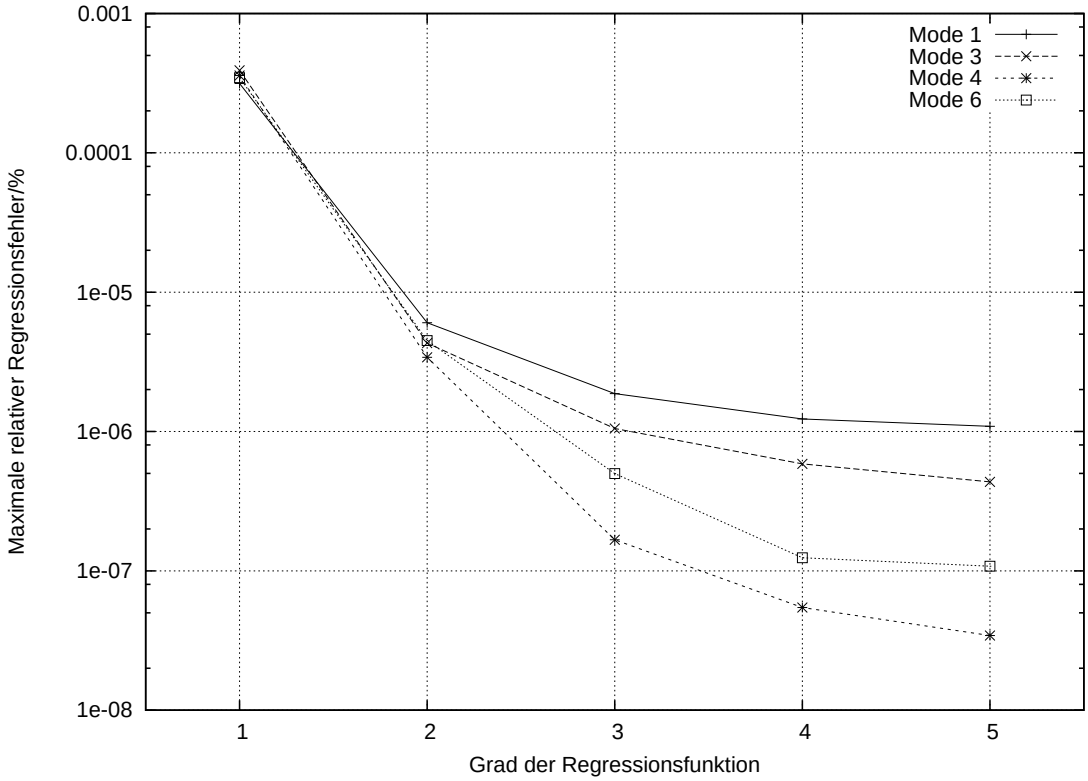


Abbildung 4.7: Regressionsfehler in Anhängigkeit des Regressionsgrades

4.2.8 Komponente „Bewertung“

Das Prinzip der Bewertung einer TS ist in Abschnitt 3.2 erläutert. Die Umsetzung der einzelnen Bewerter ist objektorientiert. Ein Basisbewerter besteht aus den Bestandteilen *Beschreibung*, *Vektor der Bewertungen*, *Vektor der Unterbewerter*, sowie *Bewertungsfunktion*. Zusätzlich besitzt er statische Funktionen für die Ermittlung der Baumtiefe und das rekursive Durchlaufen des Bewertungsbaumes. Das sind Funktionen, die für jede Klasseninstanz gleich sind. Zu bewertende Daten werden durch die Unterbewerter oder einen Datenvektor bereitgestellt.

Aus dem Basisbewerter können durch Vererbung angepasste Bewerter erzeugt werden. In dieser Arbeit sind das die in Abschnitt 3.2 vorgestellten Bewerter für Messbarkeit, Eignung, Frequenzen, Eigenmodeformen, Sensitivitäten der Parameter und Koeffizienten der Antwortflächen.

Die einzelnen Bewerter werden über die Bewertungsfunktion verbunden. Diese kann eine analytische Funktion sein oder eine Bewertung auf Basis von Intervallen bereitstellen.

Die Bewertung auf Basis von Intervallen erfolgt mit zwei Vektoren — dem Intervallvektor $\vec{I}$und dem Intervallbewertungsvektor $\vec{B}$. Der Intervallvektor $\vec{I}$ enthält

n Intervallgrenzen g_i (siehe Gleichung 4.10).

$$\vec{I} = (g_1, g_2, \ldots, g_n)^T \tag{4.10}$$

Sie bilden die Intervalle in Tabelle 4.5. Datenwerten p, welche innerhalb der Intervalle liegen, werden mit der entsprechenden Note e_i des Intervallbewertungsvektors (siehe Gleichung 4.11) bewertet.

$$\vec{B} = (e_1, e_2, \ldots, e_{n+1})^T \tag{4.11}$$

Der Intervallbewertungsvektor ist stets ein Element länger als der Intervallvektor. Mit diesem zusätzlichen Element wird das Intervall gegen $+\infty$ bewertet.

Tabelle 4.5: Intervalle und Bewertungen eines Parameter p

Intervall			Bewertung von p
$-\infty <$	p	$\leq g_1$	e_1
$g_1 <$	p	$\leq g_2$	e_2
	$\vdots$		$\vdots$
$g_n <$	p	$< +\infty$	e_{n+1}

Nach diesem Schema sind beliebige Bewertungen umsetzbar. Kontinuierliche und intervallbasierte Bewerter sind Spezialformen des Basisbewerters und bereits vorgefertigt. Alle relevanten Klassen beginnen mit `Evalution`. Für die Optimierung in dieser Arbeit wurde die in Abschnitt 4.1 beschriebene Bewertung realisiert.

4.2.9 Komponente „HTML-Ausgabe“

Die Zusammenstellung von Teststrukturdaten aus der Bewertung beziehungsweise aus der Optimierung mit dem GA erfolgt in der Auszeichnungssprache HTML [MK07]. Der Grund ist die bessere Übersichtlichkeit im Vergleich zu den ASCII-Dateien, welche zusätzlich vorhanden sind.

Die Klasse `Build_Directory_Hierarchie` baut eine Dateihierarchie nach Abbildung 4.8 auf. Für jede Teststruktur wird ein Ordner angelegt, welcher Unterordner für Eigenmodenbilder, Logdateien und die HTML-Dateien beinhaltet. Die Wurzel ist bei der Verwendung des GA ein Generationsordner, ansonsten ein beliebiger Hauptordner.

Die Klasse `Generate_Html_Summary` erzeugt für jede Teststruktur HTML-Seiten mit dem Titeln *Allgemein*, *Eigenmoden*, *Parametervariation*, *Regression* und *Bewertung*. Die Seite *Allgemein* enthält allgemeine Informationen über die Teststruktur, wie deren Bezeichnung und ihre Abmaße, sowie ein Bild der Geometrie. Auf der Seite *Eigenmoden* werden Eigenfrequenzen und Bilder der Eigenmoden der Normstruktur dargestellt, die variierten Parameter und deren Werte finden sich auf der Seite *Parametervariation*. Die Seite *Regression* fässt Informationen zur Regression, die Regressionskoeffizienten und -fehler zusammen. Die Bewertung einer

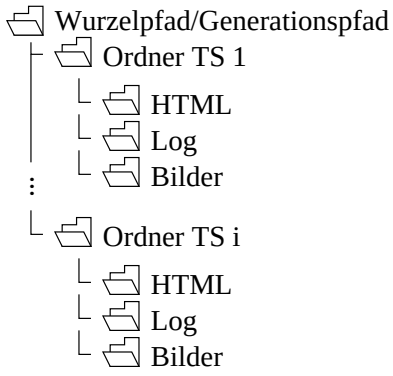


Abbildung 4.8: Dateihierarchie für die Teststrukturen

Teststruktur wird auf der Seite *Bewertung* dargestellt. Die HTML-Seiten werden im HTML-Ordner der Teststruktur abgelegt.

Zur gesammelten Anzeige mehrerer Teststrukturen wird eine HTML-Seite, namens `index.html` mit zwei Frames verwendet. Einer dient zur Navigation innerhalb einer Baumansicht der Teststrukturen, im anderen werden die entsprechenden HTML-Seiten der ausgewählten Teststruktur dargestellt. Die Datei `index.html` wird durch die Klasse `generate_index_html` generiert. Die Baumansicht wird über eine JavaScriptfunktion realisiert und durch die Klasse `generate_tree_view` in der Datei `treeview.html` abgelegt.

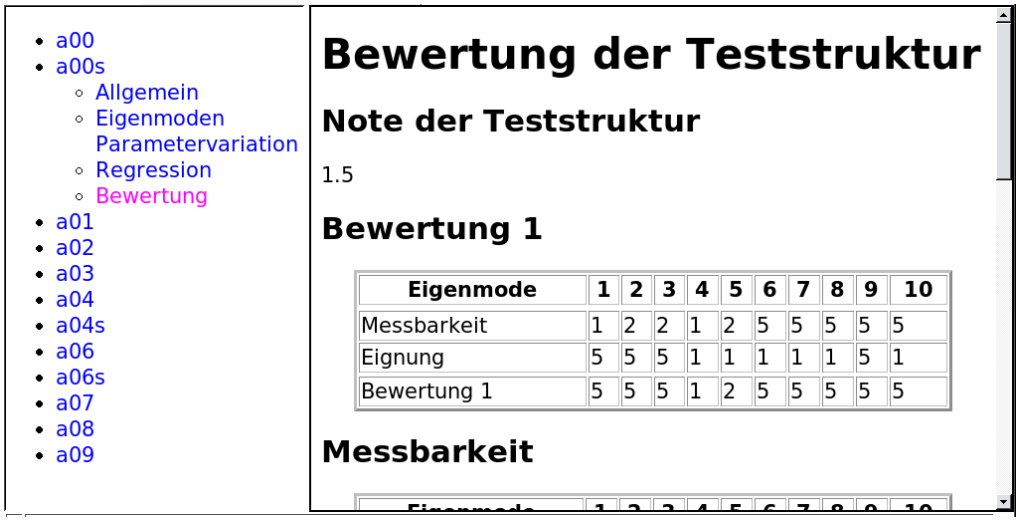


Abbildung 4.9: Browseransicht der HTML-Ausgabe

4.3 Umsetzung der Keiligkeit in ANSYS

Die Umsetzung des in Abschnitt 3.5.2 beschriebenen Vorgehens gliedert sich in zwei Aufgaben. Zuerst werden die Richtungskodes für jeden Punkt der Außenkontur

ermittelt und gesichert. Im zweiten Schritt wird aus diesen Daten die Keiligkeit berechnet.

Die Temperaturgradienten werden über eine stationäre Temperaturfeldberechnung bestimmt. Das verwendete finite Element ist das thermische 2-D-Element `PLANE77` in Abbildung 4.10. [ANS]

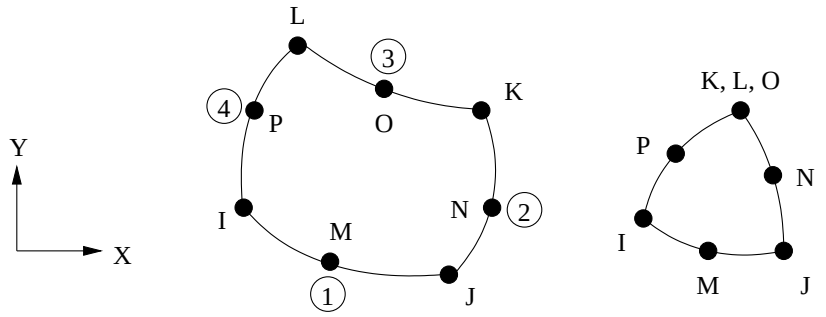


Abbildung 4.10: Geometrie von PLANE77 [ANS]

Das Element ist durch acht Knoten und orthotropische Materialeigenschaften definiert. Bei einem dreieckigen Element fallen die Knoten `K`, `L` und `O` zusammen. Lasten können an den mit 1 bis 4 markierten Knoten angreifen. Es können Materialparameter für die Dichte `DENS`, spezifische Wärme `C`, Enthalpie `ENTH` und die Wärmeleitfähigkeit in x-Richtung `KXX` und in y-Richtung `KYY` angegeben werden. In dieser Simulation wird die Wärmeleitfähigkeit verwendet. [ANS]

In Abbildung 4.11 ist der Programmablaufplan für die Ermittlung der Richtungskodes dargestellt. Es wird die Grundfläche der Struktur betrachtet. Im Präprozessor wird der Elementtyp `PLANE77` gewählt und die Fläche vernetzt. Im Lösungsprozessor wird die Außenkontur der Struktur auf die Temperatur $T_1 = 0$ gesetzt, die restliche Struktur auf eine höhere Temperatur $T_2 = 1$. Dies sind die Randbedingungen für die Temperaturfeldanalyse. Nach dem Durchführen dieser werden die Knotenlösungen im Postprozessor angezeigt. Alle Punkte der Außenkontur werden zur Komponente `kptm` gruppiert. Es wird ein Feld `kp_array` mit zwei Spalten und je einer Zeile für die Punkte in der Komponente `kptm` erzeugt. Für jeden Punkt i werden die Komponenten t_x und t_y des Temperaturgradienten eines im Medium liegenden Knotens betrachtet. Sie werden als Null angenommen, wenn ihr Betrag kleiner als $0,001$ ist. Anhand dieser wird der Richtungskode nach Tabelle 3.5.2 ermittelt. Die Punktnummer wird in `kp_array[i,1]`, der Richtungskode in `kp_array[i,2]` gespeichert. Sind alle Punkte verarbeitet, wird das Feld in einer Datei gespeichert.

Für die Anwendung der Unterätzung auf diese Struktur ist nur noch die Datei mit Punktnummern und Richtungskodes notwendig. Der Programmablaufplan ist in Abbildung 4.12 dargestellt. Dieser Programmteil wird nach dem Aufbau der Geometrie der Struktur und vor dem Anlegen der Randbedingungen eingefügt. Die Datei mit den Richtungskodes wird in ein Feld eingelesen. Dieses beinhaltet die zu verschiebenden Punkte. Für jeden Punkt in diesem Feld werden die aktuellen Koordinaten ausgelesen. Anhand des zugehörigen Richtungskodes wird der Punkt

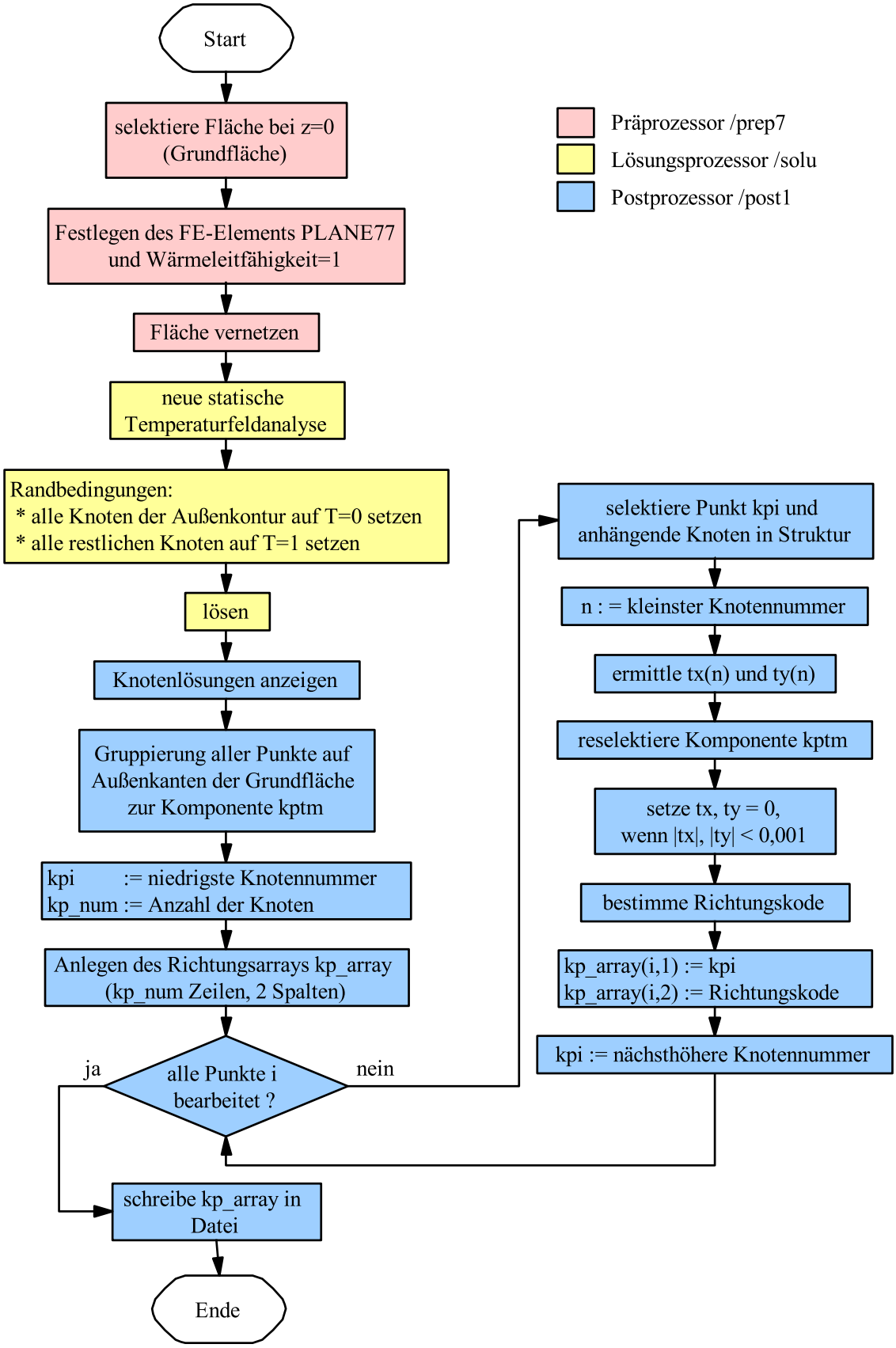


Abbildung 4.11: PAP Umsetzung der Keiligkeit in ANSYS

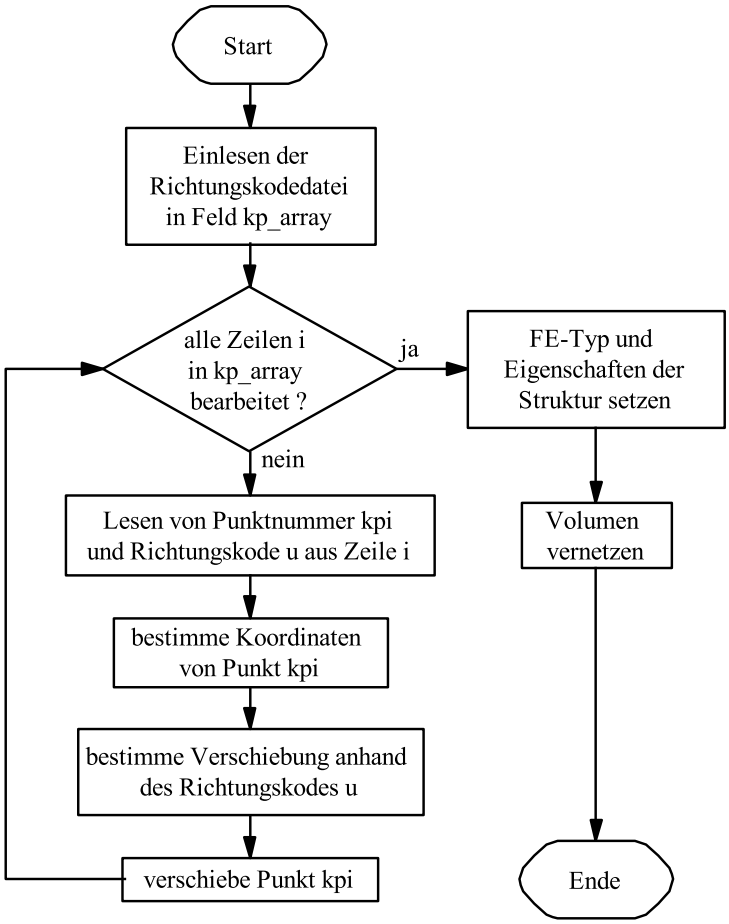


Abbildung 4.12: PAP Anwendung der Keiligkeit

mit dem Befehl `kmodif` entsprechend Tabelle 3.5.2 um Δu_x in x-Richtung und Δu_y in y-Richtung verschoben. Anschließend wird der für die Simulation benötigte Elementtyp, seine Form und die Größe gewählt und das Volumen vernetzt.

Abbildung 4.13 zeigt einen Ausschnitt aus einer Struktur und die Temperaturgradienten in Vektordarstellung. In Abbildung 4.14 ist ein Ausschnitt der Struktur nach erfolgter Verschiebung veranschaulicht. Die Grundfläche ist in diesem hell und die schrägen Seitenwände sind dunkler.

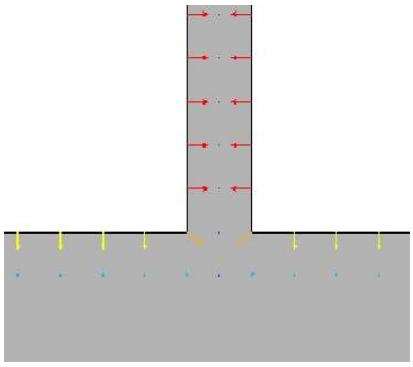


Abbildung 4.13: Strukturausschnitt mit Verschiebungsvektoren

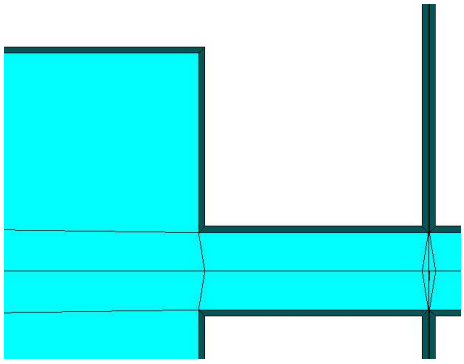


Abbildung 4.14: Strukturausschnitt umgesetzte Keiligkeit

Dieses Verfahren hat den Vorteil, dass die Richtungen der Verschiebung nur einmal bestimmt werden müssen. Weiterhin ist die Reihenfolge der Numerierung der Punkte nicht relevant. Sie muss lediglich bei der Verschiebung mit der Numerierung in der Richtungsermittlung übereinstimmen. Das entscheidende Kriterium für die Verwendung in dieser Arbeit ist, dass diese Verfahren für jede Struktur anwendbar sind und keine manuelle Bearbeitung notwendig ist. Die Schwankungen der Keiligkeit $x = u(th)$ sind nicht berücksichtigt. Dazu müsste der Verschiebealgorithmus angepasst werden, die Richtungsermittlung bleibt gleich.

4.4 Eigenmodenerkennung

Die Eigenmodenerkennung ist notwendig, um zum einen falsch sortierte Eigenmoden aus der Simulation zu korrigieren und zum anderen sie bei der Messung anhand ihrer Auslenkung zu identifizieren (vergleiche Anhang C. Diese Arbeit konzentriert sich auf das Identifizieren der simulierten Eigenmoden.

In diesem Abschnitt soll die Umsetzung der in Abschnitt 3.7 aufgezeigten Konzepte dargestellt werden. Die manuelle Eigenmodenerkennung wird im folgenden Abschnitt kurz dargestellt. Sie kann nicht für die Automatisierung genutzt werden, bringt aber Vergleichsdaten für die Güte der Identifikation von Eigenmoden mittels Neuronalem Netz. Im darauffolgenden Abschnitt wird die Umsetzung mit einem Neuronalen Netz behandelt. Die Erläuterungen erfolgen an einer Beispielstruktur.

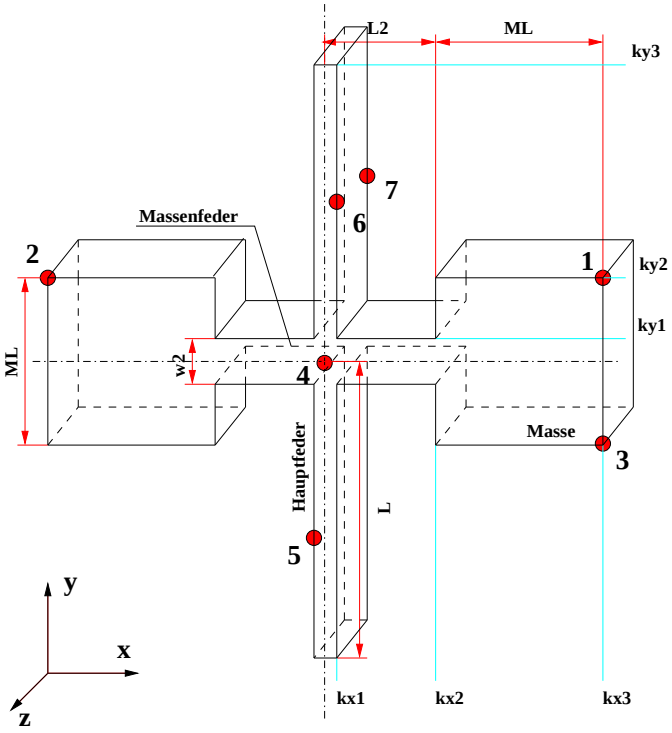
Der letzte Abschnitt befasst sich mit der Zuordnung der Eigenmoden in In-Plane und Out-of-Plane. Sie ist für die Messbarkeit von Strukturen in bestimmten

Frequenzbereichen relevant (siehe Abschnitt 2.3).

4.4.1 Manuelle Eigenmodenerkennung

Die manuelle Eigenmodenerkennung erfolgt anhand der Auslenkungen signifikanter Punkte und einem Entscheidungsbaum. Beides muss für jede Normstruktur von einem Menschen festgelegt werden. Die betrachteten Punkte werden durch Vergleich der Normeigenmoden untereinander festgelegt. Der Aufwand steigt mit der Anzahl der Eigenmoden und der Komplexität der Struktur. Besonders Eigenmoden in denen primär Auslenkungen von Federn erfolgen, sind nur mit einer erhöhten Anzahl signifikanter Punkte erfassbar.

In Abbildung 4.15 ist eine Beispielstruktur mit ihren signifikanten Punkten für zehn Eigenmoden dargestellt. Abbildung 4.16 zeigt den dazugehörigen Entscheidungsbaum.



Knoten	X	Y	Z
1	kx3	ky2	p2
2	-kx3	ky2	p2
3	kx3	-ky2	p2
4	0	0	p2
5	0	-ky3/2	p2
6	0	ky3/2	p2
7	0	ky3/2	0

Abbildung 4.15: Signifikante Knoten zur Identifikation der Eigenmoden

4.4.2 Eigenmodenerkennung mittels Neuronalem Netz

Das Neuronale Netz hat bei der Modenerkennung die Aufgabe, die Auslenkung einer Struktur mit einem Eigenmode in Verbindung zu bringen. Es dient somit im weitesten Sinne als Assoziativspeicher. Der abstrahierte Aufbau ist in Abbildung 4.17 dargestellt. Prinzipiell kann das Neuronale Netz durch verschiedene Netzwerkmodelle repräsentiert werden. In dieser Arbeit wird das in Abschnitt 2.9.4 vorgestellte

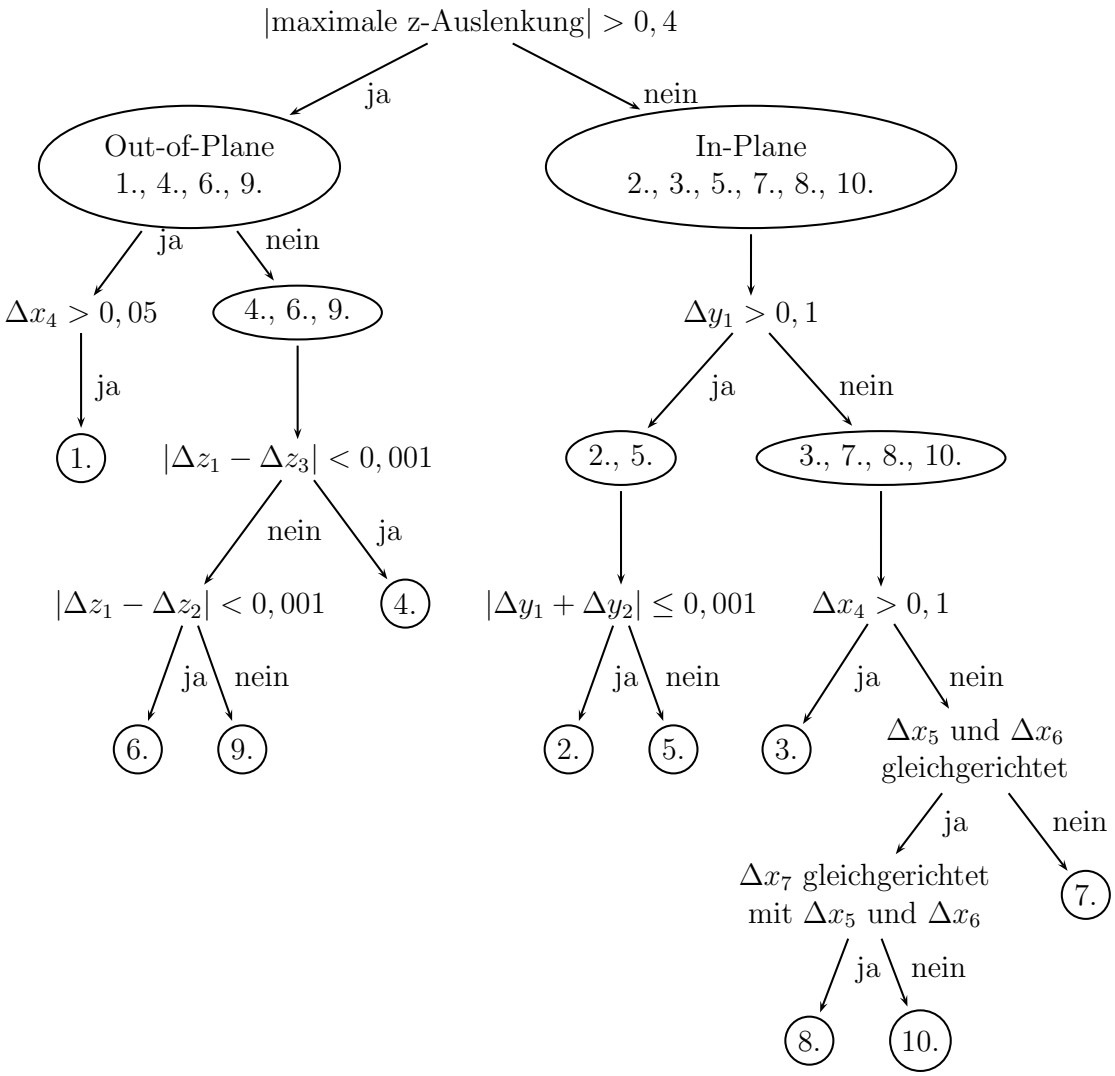


Abbildung 4.16: Entscheidungsbaum für manuelle Eigenmodenerkennung

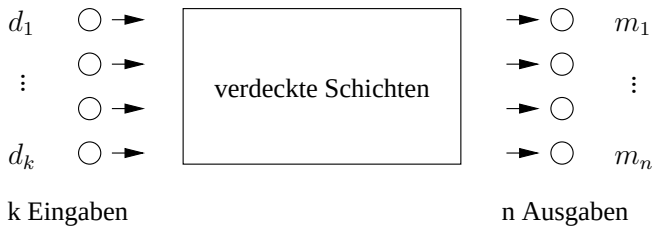


Abbildung 4.17: Aufbau des NN zur Modenerkennung

BPN verwendet. Das Konzept der Eigenmodenerkennung mittels NN wurde in Abschnitt 3.7 vorgestellt.

Als Eingabedaten d_k werden die Auslenkungen der betrachteten Struktur an den Koordinaten P_i bei einer Eigenfrequenz genutzt. Die Ausgabedaten beim Training sind eine Kodierung des Eigenmodes. Die Anzahl n der Ausgabeneuronen m

entspricht der Anzahl der betrachteten Eigenmoden. Für eine bestimmte Eigenmode wird das ihrer Nummer entsprechende Ausgabeneuron auf 1 gesetzt, die anderen auf -1 (siehe Gleichung 4.12).

$$o_i = \begin{cases} m_j & = & 1 \\ m_{i \neq j} & = & -1 \end{cases} \forall \text{ Eigenmoden j und Ausgabeneuronen i} \tag{4.12}$$

Die Trainingsdaten werden durch die Modalanalyse aus ANSYS extrahiert und in der entsprechenden Formatierung in die Trainingsdatei gespeichert. Der Aufbau ist in Listing 4.1 dargestellt, welches sich an Listing 2.1 orientiert. Beispielhaft sind die Ein- und Ausgabedaten für die ersten und letzten Eigenmode aufgezeigt.

Listing 4.1: Trainingsdatenformat

<# Eigenmoden n> <# Auslenkungen k> <# Eigenmoden n>
<Auslenkung d_1 Mode 1> .. <Auslenkung d_k Mode 1>
1 -1 ... -1
...
<Auslenkung d_1 Mode n> .. <Auslenkung d_k Mode n>
-1 -1 ... 1

Zur Ausführung werden ausschließlich die Eingabedaten und die jeweilige Eigenfrequenz benötigt. Die Datei beinhaltet zusätzlich weitere Daten in der ersten Zeile (bei der Parametervariation mit ANSYS das zugehörige Parameterset). Anstelle der Kodierung steht die dieser Auslenkung entsprechende Eigenfrequenz. Die Erstellung der Daten erfolgt über die in Abschnitt 4.2.5 beschriebene Parametervariation.

Für die Extraktion der Auslenkung aus ANSYS gibt es mehrere Möglichkeiten. Das FE-Modell in ANSYS besteht aus Punkten, Linien, Flächen, Volumen und finiten Elementen mit ihren Knoten. Es wäre möglich ein Gitter über die Struktur zu legen, jedoch besteht dabei die Gefahr, dass, wenn das Gitter zu grob ist, wichtige Strukturteile, wie beispielsweise Federn, nicht erfasst werden. Wird es zu fein gewählt, steigt die Datenmenge und damit auch der Umfang und die Verarbeitungszeit des Neuronalen Netzes.

In Tabelle 4.6 sind alternative Möglichkeiten aufgeführt, um sinnvolle Koordinaten für die Auslenkungen zu erhalten. Wie deutlich wird, ist ein Kompromiss zwischen der Datenmenge und der Abdeckung der Federn mit Datenpunkten notwendig.

In Tabelle 4.7 sind die Möglichkeiten zur Auslenkungscharakterisierung aufgeführt. Die Auslenkung kann in der Richtung wechseln, somit sind die Beträge zu betrachten. Soll das Neuronale Netz zur Identifizierung gemessener Eigenmoden verwendet werden, so sind die Koordinaten der Auslenkungen der Simulation denen der Messung anzupassen.

Auf Grund von Abweichungen in den Ausführungsdaten ist das Neuronale Netz unter Umständen nicht in der Lage, Eigenmoden korrekt zu erkennen. Federn und

Tabelle 4.6: Datenpunkte für die Eingabedaten des NN

Koordinaten	Bemerkungen
Alle Knoten einer Struktur	Zu viele Daten
Alle Punkte	Eventuell zu ungenau, Federn unzureichend abgedeckt
Alle Knoten auf einer Ebene	Zu viele Daten
Alle Knoten auf allen Linien	Viele Daten, Federn abgedeckt
Alle Knoten auf den Linien einer Ebene	Wenig Daten, Federn unzureichend abgedeckt
Alle Knoten an den Punkten einer Ebene	Wenig Daten, Federn unzureichend abgedeckt

Tabelle 4.7: Verschiebungen als Eingangsdaten eines NN

Daten pro Koordinate	Anzahl Eigabeneuronen pro Koordinate	Bemerkungen
$\Delta x, \Delta y, \Delta z$	3	• Fehler bei der Erkennung durch Richtungsumkehr möglich • sehr viele Werte
$\mid \Delta x \mid, \mid \Delta y \mid, \mid \Delta z \mid$	3	• Richtungsumkehr beachtet • sehr viele Werte
$\Delta x^2 + \Delta y^2 + \Delta z^2$	1	• Richtungsumkehr beachtet • wenige Werte

Teile mit hoher lokaler Änderung der Auslenkung, auf denen wenige Detektionspunkte liegen, sind besonders kritisch. Für diese Fälle gibt es, neben der Lösung ein anderes Netz zu verwenden, zwei weitere. Zum Einen könnten mehr Datenpunkte beziehungsweise ein dichteres Gitter verwendet werden. Damit könnten auch feinere Strukturen erfasst werden. Die Erfassung ist allerdings nicht garantiert und der Zeitaufwand des Neuronalen Netzes steigt. Zum Anderen könnte die uneindeutige Ausgabe markiert und das Parameterset für ungültig erklärt werden. Damit ließen sich Fehler vermeiden, aber es muss sichergestellt werden, dass für die Regression der Antwortfläche ausreichend korrekte Parametersets zur Verfügung stehen (siehe Anhang, Abschnitt A.1). Im Falle nur einer nicht eindeutigen Ausgabe, kann diese als die übrige Eigenmode detektiert werden.

Das Neuronale Netz selbst ist von der zufälligen Initialisierung der Gewichte beim Training abhängig. Damit entsteht bei jedem Training ein anderes Netz. Dadurch kommt es zu Schwankungen in der Güte der Erkennung. Die Güte der Erkennung ist weiterhin von den Netzeinstellungen und von der Interpretation der Ausgabe des Neuronalen Netzes abhängig. Weitere Ausführungen, sowie Beispieldemonstrationen finden sich in Abschnitt 5.1.

4.4.3 Zuordnung der Eigenmodenform

Die Erkennung der Hauptauslenkungsrichtung ist für die Messung relevant. Generell können Out-of-Plane Moden besser gemessen werden als In-Plane Moden (siehe Abschnitt 2.3).

Für die Ermittlung der Hauptauslenkungsrichtung in der Simulation wird nach einer Modalanalyse eine Eigenmode ausgewählt. Für diese wird die Knotenverschiebung in z-Richtung angezeigt.

Ist der maximale absolute Wert der Knotenverschiebung größer als $0,4$, so wird die Eigenmode als Out-of-Plane erkannt, ansonsten als In-Plane (siehe Tabelle 4.8). Die Identifikation im Programm erfolgt über den zugewiesenen Schlüssel.

Tabelle 4.8: Zuordnung der Bewegungsrichtung

Bewegung	In-Plane	Out-of-Plane
$\lvert u_{z\max}\rvert$	$\leq 0,4$	$> 0,4$
Schlüssel	1	0

4.5 Betrachtete Geometrien von Teststrukturen

In dieser Arbeit wurden zwölf Grundgeometrien für Teststrukturen entwickelt und untersucht. Diese sind nicht optimiert, bieten aber eine Grundlage dafür. In Anhang D sind alle Geometrien und ihre ersten zehn Eigenmoden dargestellt. Mit Ausnahme der Struktur `a08` sind alle aufgeführten viertelsymmetrisch.

Kapitel 5

Ergebnisse

In diesem Kapitel werden die Ergebnisse der Umsetzung von Kapitel 4 aufgezeigt und diskutiert. An die Ausführungen zur Eigenmodenerkennung schließt sich die Bewertung von untersuchten Teststrukturen an. Daraufhin wird diese Bewertung mit durchgeführten Messungen verglichen. Abschließend erfolgen Bemerkungen zur Teststrukturoptimierung.

5.1 Eigenmodenerkennung

Die Ergebnisse der Eigenmodenerkennung werden in den folgenden Abschnitten behandelt. Die Erkennung mittels Neuronalem Netz wurde auf dem in Tabelle 5.1 spezifizierten Testsystem durchgeführt.

Tabelle 5.1: Spezifikation des Testsystems zur Eigenmodenerkennung mit NN

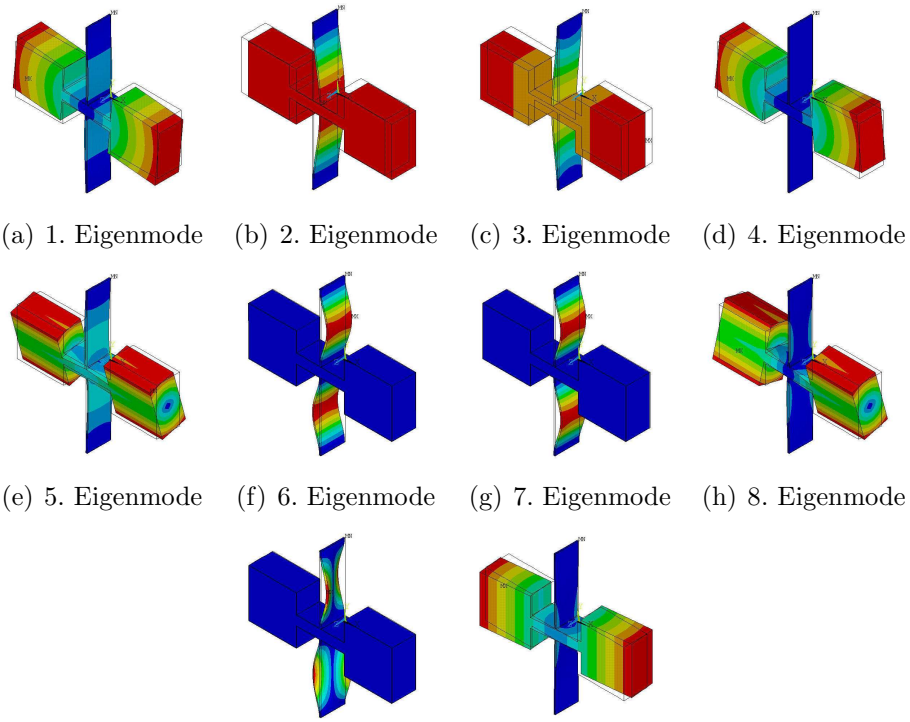
Eigenschaft	Wert
Prozessor	AMD Athlon(tm) 64 Processor 2800+
Taktfrequenz	1802 MHz
Datenbreite	32 Bit
Arbeitsspeicher	1 GB
Betriebssystem	Linux (Kernel: 2.6.26.2)

Der folgende Abschnitt behandelt die Eigenmodenerkennung mittels Neuronalem Netz in Abhängigkeit verschiedener Einflussfaktoren. Im darauffolgenden werden die Auswirkungen von unsortierte, manuell und mit einem Neuronalen Netz sortierten Eigenmoden auf den Regressionsfehler dargelegt.

5.1.1 Modenerkennung mit NN

Bei der Modenerkennung mittels NN wird zuerst durch eine Modalanalyse mit dem Normparameterset eine Trainingsdatei für eine Teststruktur (hier TS1) erstellt. Da-

bei werden die summierten Auslenkungen an Knoten bei der jeweiligen Eigenmode abgespeichert. Für jede Eigenmode wird so ein charakteristisches Auslenkungsbild gewonnen. Mit diesen Daten wird das Neuronale Netz trainiert. Die Testdaten für die Parametervariation werden für jedes Parameterset analog abgespeichert. Die Auswertung über das Neuronale Netz ergibt Wahrscheinlichkeiten, mit der bei einem Auslenkungsbild ein Eigenmode vorliegt. Auf Grundlage dessen können die Eigenfrequenzen sortiert werden. Abbildung 5.1 zeigt die ersten zehn Eigenmoden für das Normparameterset.



(a) 1. Eigenmode (b) 2. Eigenmode (c) 3. Eigenmode (d) 4. Eigenmode

(e) 5. Eigenmode (f) 6. Eigenmode (g) 7. Eigenmode (h) 8. Eigenmode

(i) 9. Eigenmode (j) 10. Eigenmode

Abbildung 5.1: Eigenmoden des Normparametersets von TS1

In dieser Testreihe wurde ein Neuronales Netz (BPN) mit drei Schichten und 17 verdeckten Neuronen verwendet. Das Netz an sich bedarf weiterer Optimierung. Bei einem Fehler kleiner als 0.001 gilt das Netz als trainiert. Als Aktivierungsfunktion wird eine symmetrische Sigmoid-Funktion genutzt.

Die Erkennung der Eigenmoden erfolgt anhand der jeweils größten Ausgabe-Werte des NN bei den Auslenkungsbilder. Ein Parameterset gilt als erkannt, wenn alle Eigenmoden eindeutig zugeordnet wurden konnten.

Die Testreihe bestand aus fünf Tests, welche die summierten Auslenkungen an unterschiedlichen Knoten betrachten (siehe Tabelle 5.2).

An jeder Testreihe wurde die Modenerkennung mittels NN 100mal für 24 Parametersets durchgeführt. Da jedes trainierte NN anders ist und damit eine andere Güte hat, kann somit eine Häufigkeitenanalyse durchgeführt werden. Zusätzlich wurde jede Testreihe mit sechs, sieben, acht, neun und zehn Eigenmoden simuliert.

Tabelle 5.2: Tests und betrachtet Knotenauslenkungen

Test	Beschreibung	Eingangsdaten NN
u_displ1	Knoten auf allen Linien des Modells	621
u_displ2	Knoten auf der Grundfläche des Modells	244
u_displ3	Knoten an allen Linien der Grundfläche des Modells	258
u_displ4	Knoten an allen Punkten des Modells	70
u_displ5	Knoten an den Punkten der Grundfläche des Modells	35

Abbildung 5.2 zeigt die mittleren Erkennungshäufigkeiten in den einzelnen Tests bei verschiedenen Eigenmoden. Dabei wird deutlich, dass bei wenigen Eigenmoden die Häufigkeit der Erkennung der Parametersets steigt, während die Anzahl der Eingangsdaten tendenziell sinkt. Je mehr Eigenmoden betrachtet werden, desto schlechter wird die Erkennung. Weiterhin ist der am besten geeignete Test von den betrachteten Eigenmoden abhängig.

Die Erklärung findet sich in den Eigenmoden selbst. Treten keine Federmoden auf, so liefert eine Erkennung mit wenigen Eingangsdaten gute Werte. Mit steigender Anzahl Eigenmoden nehmen die auftretenden Federmoden zu. Damit decken die Daten von `udispl5` diese nicht ausreichend ab. Werden mehr Eingangsdaten verwendet, so verbessert sich die Erkennung. Da die Datenpunkte generisch ausgewählt werden, gibt es keine Garantie, dass die Feder optimal erfasst werden. Weisen höhere Eigenmoden Ähnlichkeiten mit niedrigeren auf, wie Eigenmoden 1 und 10 (siehe Abbildung 5.1), so kann die Erkennung ebenfalls fehlerbehaftet sein.

Die mittleren Erkennungshäufigkeiten sind nur als Orientierung anzusehen. In Anhang E sind die Häufigkeitsverteilungen der einzelnen Tests aufgeführt. Generell treten konzentrieren sich die Hauptanteile auf eine vollständig beziehungsweise eine komplett fehlgeschlagene Erkennung. Mit steigender Anzahl Eigenmoden steigt auch hier der Anteil der Fehlerkennung.

In der Testreihe traten Fehler in der Erkennung der Moden auf, diese können sich ergeben aus:

- unguenstig gewaehlten Messpunkte in der Struktur
- Knotenverschiebungen auf Grund von Keiligkeit
- Knotennummernverschiebungen
- einer unzureichenden Sortierfunktion/Interpretation des NN
- einem ungeeigneten NN

Es wurde nur die summierte Verschiebung der jeweiligen Knoten berücksichtigt, es ist zu prüfen, ob wenige Knoten und getrennte Betrachtung der X-, Y- und Z-

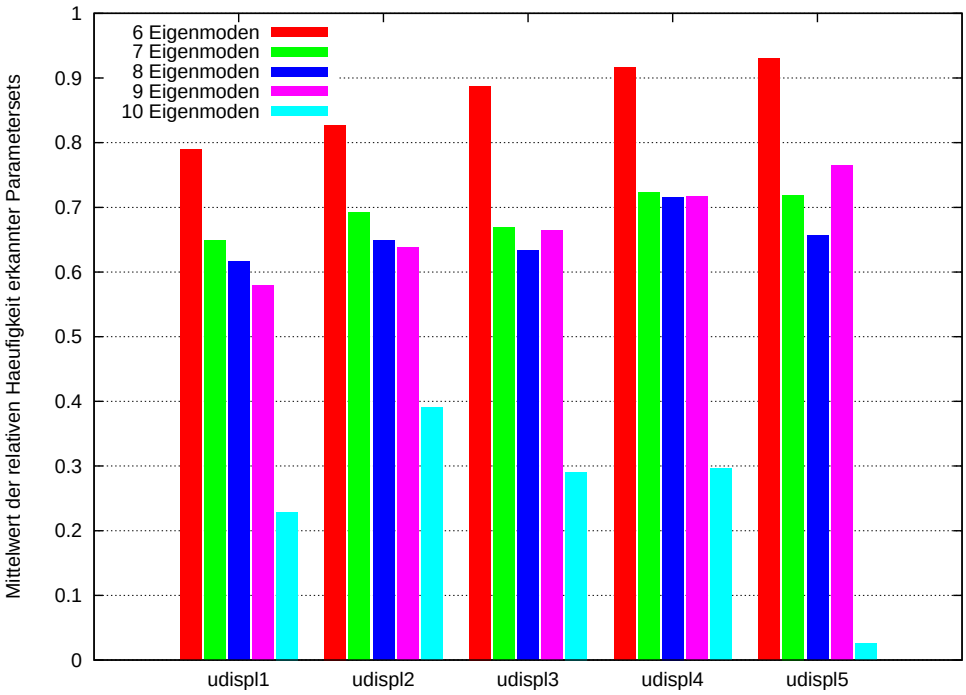


Abbildung 5.2: Mittlere Erkennungshäufigkeiten bei verschiedenen Eigenmoden

Verschiebung bessere Ergebnisse bringt. Weiterhin ist festzuhalten, daß die Ausführungszeit des Neuronalen Netzes mit steigender Knotenzahl größer wird (siehe Abbildung 5.3. Fehlerhafte oder zu viele Daten verhindern unter Umständen das Konvergieren der Netzes. Die Trainings- sowie Ausführungszeit eines Parametersets liegt im Durchschnitt unter einer Sekunde.

5.1.2 Vergleich manuelle und neuronale Modenerkennung

In diesem Abschnitt werden Regressionsfehler bei linearer Regression für verschiedene Eigenmodensortierung verglichen.

Abbildung 5.4 zeigt den prozentualen maximalen relativen Regressionsfehler für unsortierte, manuell und mit dem Neuronen Netz aus Abschnitt 5.1.1 sortierte Eigenmoden. Für viele Eigenmoden ist er nahezu gleich. Bei den Eigenmoden 4 und 5 tritt ein höherer Fehler auf. Sie tauschen im Verlauf der Parametervariation. Da die unsortierte Variante und die manuell sortierte einen hoher Fehlerwert im Vergleich zum neuronal sortierten aufweisen, liegt die Vermutung nahe, dass die manuelle Modenerkennung nicht immer zuverlässig funktioniert. Die Amplitudenschwellwerte, an denen entschieden wird, welcher Eigenmode vorliegt, sind für die gesamte Parametervariation schwer einzuschänken. Besonders wenn die Schwellwerte nur einem Teil der Variation entnommen sind, kann es zu Fehlern in der Eigenmodenerkennung kommen, wenn die Parameter über größere Bereiche variiert werden. Bei der Erkennung mittels Neuronalem Netz liegt der maximale relative Regressionsfehler deutlich niedriger. Der vorhandene Fehler kann auf Fehler im

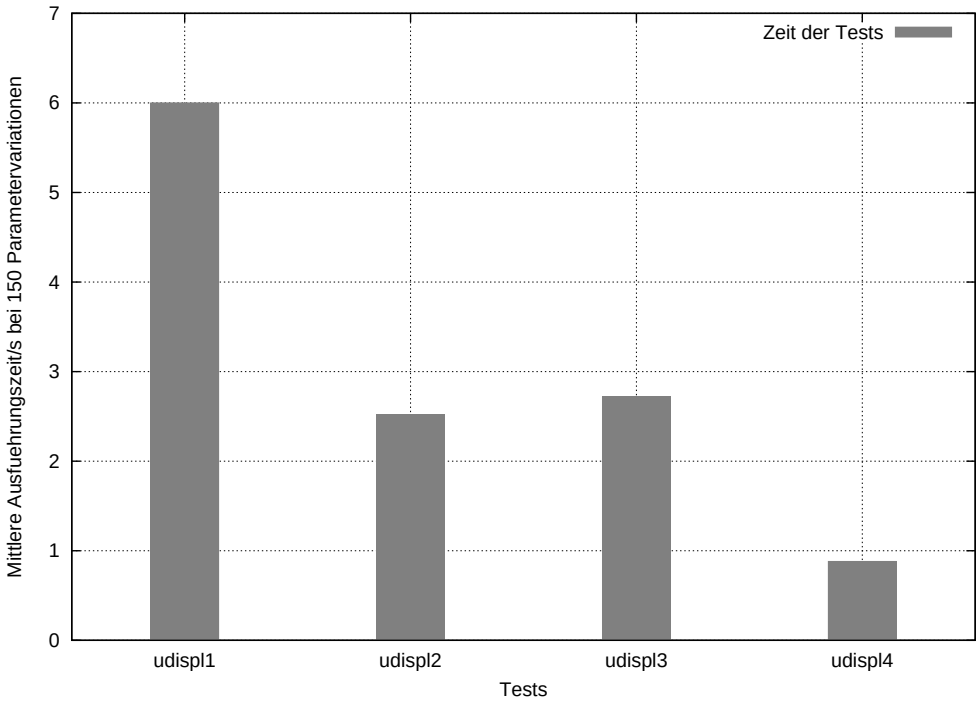


Abbildung 5.3: Mittlere Ausführungszeit des NN über 150 Parametersets

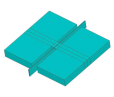
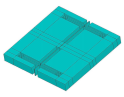
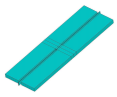
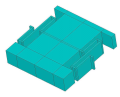
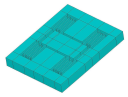
Neuronalen Netz (siehe Abschnitt 5.1.1) und eine ungünstige Regressionsfunktion zurückgeführt werden.

Die zehnte Eigenmode wurde von beiden Erkennungsverfahren schlecht erkannt. Die Ursache liegt in der Ähnlichkeit zur ersten Eigenmode.

5.2 Bewertung der Teststrukturen

Auf die Teststrukturgeometrien aus Anhang D ist das Bewertungsschema aus Abschnitt 4.1 angewand worden. Die Endnoten einer Auswahl sind in Tabelle 5.3 aufgeführt. Die Noten aller Strukturen finden sich in Anhang D.

Tabelle 5.3: Bewertung der Teststrukturen (Auswahl)

Teststruktur	a00	a01	a02	a06	a08	a09
Geometrie						
Note	2	5	2	2	3	2,5

Bei der Bewertung wurden die Parameter Strukturdicke th und die Keiligkeit x unter Verwendung einer linearen Regressionsfunktion betrachtet. Die Bewertung

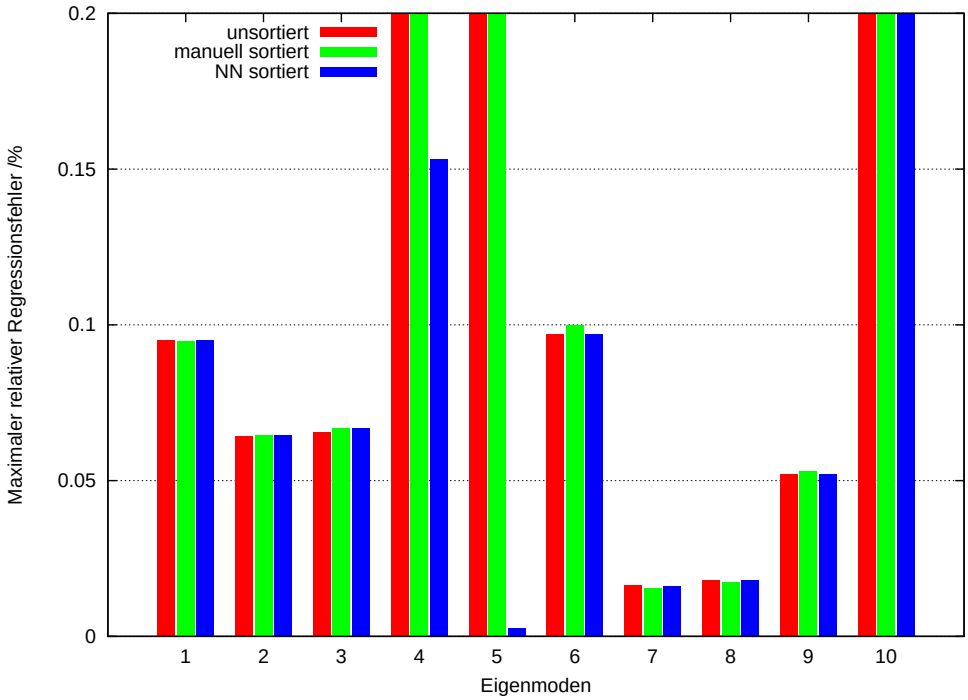


Abbildung 5.4: Vergleich der maximalen relativen Regressionsfehler

gibt allerdings keine Auskunft darüber, für welchen Parameter eine Struktur zu bevorzugen ist.

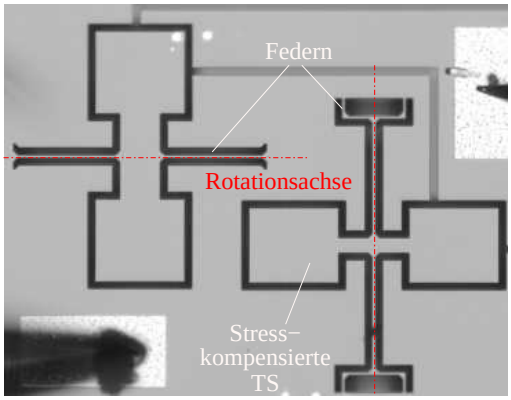
Wie erwartet, erhielt der einseitig eingespannte Biegebalken `a01` die schlechteste Bewertung. Seine Eigenfrequenzen liegen fast ausschließlich im nichtmessbaren Bereich.

Teststruktur `a08` wird als befriedigend eingestuft, obwohl ihr Sensitivität gegenüber beiden Parameter bei den meisten Eigenmoden größer ist als bei anderen Strukturen. Da ihre Eigenfrequenzen insgesamt aber höher liegen und in den messbaren Frequenzen der Sensitivitätsunterschied der Parameter gering ist, erfährt sie eine Abwertung.

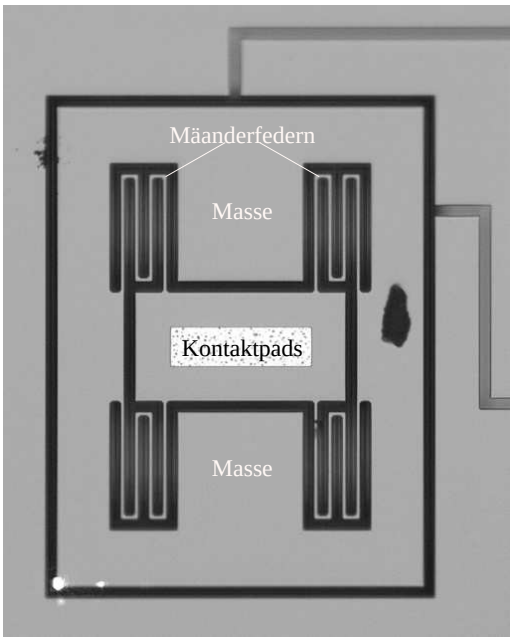
Weiterhin wird deutlich, daß sich große Masseflächen positiv auf das Verhalten auswirken — vergleiche `a00`, `a02` und `a06`. Dies kann mit der Lösung von Gleichung 2.5 begründet werden. Die Teststruktur `a06` hat, verglichen mit den restlichen Teststrukturen, wesentlich längere Federn und Masseflächen. Struktur `a09` zeigt ebenfalls für die Parameteridentifikation ein günstiges Verhalten. Sie weist mehr Eigenfrequenzen im messbaren Bereich auf als beispielsweise `a00`. Dies schlägt sich nicht in dieser Bewertung nieder, ist jedoch bei mehr betrachteten Fertigungsparametern relevant.

5.3 Vergleich von Bewertung und Messung

In diesem Abschnitt werden klassische und neu entwickelte Teststrukturen miteinander verglichen. Abbildung 5.5 zeigt Aufnahmen von gefertigten Teststrukturen. Die klassischen Strukturen entsprechen in diese Arbeit `a04`, sowie als stresskompsierte Variante `a04s`. Von der neuen Teststrukturgeneration wird `a09` betrachtet.



(a) Klassische Teststrukturen



(b) Neue Teststruktur

Abbildung 5.5: Aufnahmen von klassischer und neuer TS

Der Amplitudengang eines einfachen Feder-Masse-Dämpfer-Systems ist in Abbildung 5.6 dargestellt. Eigenfrequenzen werden durch Amplitudenüberhöhungen angezeigt. Die Eigenfrequenzen des einfachen Systems liegen im MHz-Bereich, es sind zwei zu erkennen. Die klassische Teststruktur zeigt vier Überhöhungen der Amplitude (siehe Abbildung 5.7. Die zugehörigen Frequenzen liegen deutlich niedrigen im kHz-Bereich. Der Amplitudengang einer neuen Teststruktur, dargestellt in Abbildung 5.8, verdeutlicht, dass sie mehr Eigenfrequenzen im messbaren Bereich besitzt. Damit können mit dieser Struktur mehrere Parameter identifiziert werden, da die Detektion der entsprechenden Eigenfrequenzen möglich ist.

Das einfache Feder-Masse-Dämpfer-System ist mit Teststruktur `a01`, die klassische Teststruktur mit `a04` und die neue Teststruktur mit `a09` gleichzusetzen. Die Ergebnisse der Bewertung der Teststrukturen aus dem vorherigen Abschnitt sind durch die Messung bestätigt.

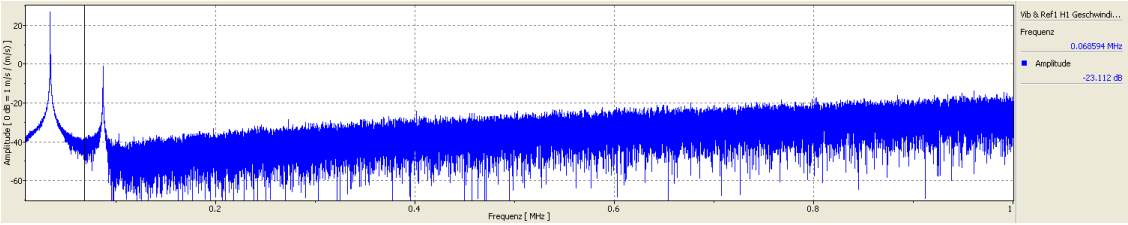


Abbildung 5.6: Amplitudengang eines einfachen SMD

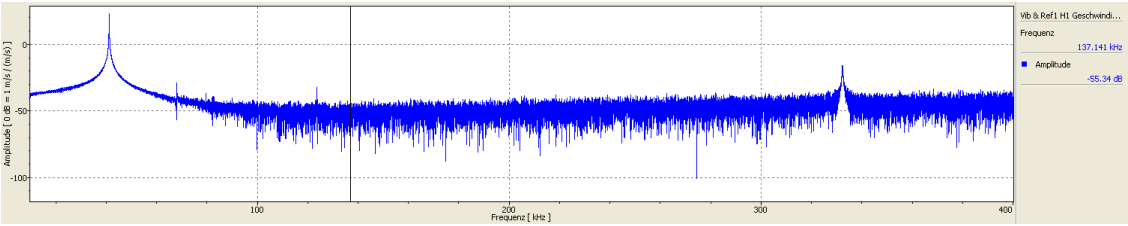


Abbildung 5.7: Amplitudengang einer klassischen Teststruktur

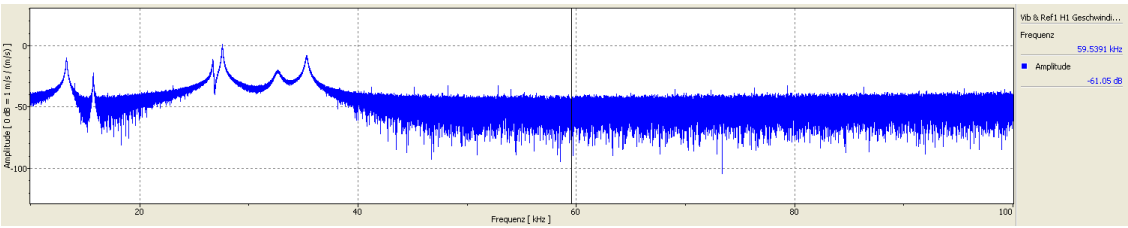


Abbildung 5.8: Amplitudengang einer neuen Teststruktur

5.4 Teststrukturenoptimierung

Die Teststrukturoptimierung mittels GA ist in Anhang G an einer Struktur demonstriert. Diese beinhaltet nicht das in dieser Arbeit beschriebene Bewertungsschema, sondern wird nach einfacheren Kriterien, wie laterale Ausdehnung und Wert der ersten Eigenfrequenz, optimiert.

Im Laufe der Arbeit hat sich herausgestellt, dass die Verwendung des in Abschnitt 4.1 in einem Genetischen Algorithmus nicht ausreichend Differenzierung liefert. Der Genetische Algorithmus konvergiert sofort. Für die Optimierung mittels GA müsste somit die Bewertung angepasst werden. Ein Punktesystem, bei welchem eine Teststruktur umso mehr Punkte erhält, je besser sie die Bewertungskriterien einhält, könnte eine feine Bewertung erzielen. Im Zuge dessen müssten auch die Bewertungsfunktionen für Messbarkeit und Eignung überarbeitet werden.

Die Hauptursache der geringen Differenzierbarkeit ist die Berechnung der Gesamtnote. Sie setzt sich aus den n besten Eigenmodennoten zusammen, die mindestens für die Identifikation der gewünschten Parameter benötigt werden. Dabei wird keine Rücksicht darauf genommen, wie gut sich die Struktur für welchen Pa-

rameter eignet. Die Eignung für bestimmte Parameter müsste somit direkt in die Gesamtwertung eingehen. Dies ist mit dem verwendeten Bewertungsbaum nicht möglich.

Wird nur die Optimierung auf einen Parameter angestrebt, so ist diese mit dem Genetischen Algorithmus nach Anpassung der Eignungsbewertung theoretisch möglich. Die Eignung müsste in diesem Fall gut bewertet werden, wenn die Trennbarkeit der betrachteten Parameter gegeben und die Sensitivitätsbewertung des zu optimierenden Parameters hoch ist.

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde die algorithmische Optimierung von Teststrukturen zur Charakterisierung von Mikrosystemen auf Waferebene untersucht. Zu diesem Zweck wurde eine Literaturstudie durchgeführt, welche zeigte, dass im Bereich MEMS bei komplexen Optimierungsaufgaben auf Evolutionäre Algorithmen, Neuronale Netze und Zelluläre Automaten zurück gegriffen wird. Die Untersuchung der theoretischen Grundlagen ergab, dass Zelluläre Automaten für die vorliegende Aufgabe nicht geeignet sind.

Zur Optimierung wurde letztlich ein Genetischer Algorithmus eingesetzt, mit welchem die Geometrie vorliegender Teststrukturen manipuliert wird. Grundlage für die Verwendung des Genetischen Algorithmus ist die Bewertung von Teststrukturen. Diese Aufgabe ist nicht trivial und breit gefächert. Die vorliegende Arbeit setzt die Bewertung in einer Baumstruktur, bei der die Grenzen des verwendeten Messgerätes, sowie teststrukturspezifische Eigenschaften eingehen, um. Damit werden Strukturen auf einer Skala von 1 bis 5 bewertet, wobei 1 der beste Wert ist. Das Ziel war, Teststrukturen zu ermitteln, die sich gut für die Parameteridentifikation eignen und zudem ein gutes Messverhalten aufweisen.

Im Zuge der Bewertung wird eine Parametervariation der Fertigungsparameter durchgeführt. Bei dieser Parametervariation kann es zum Vertauschen von Eigenmoden kommen, was sich negativ auf die Antwortfläche und damit auf die Bewertung auswirkt. Zu diesem Zweck wird ein Neuronales Netz zur Eigenmodenerkennung eingesetzt, da der Genetische Algorithmus eine manuelle Erkennung ausschließt. Es ist deutlich geworden, dass diese Art der Eigenmodenerkennung prinzipiell funktioniert, allerdings besteht an der Qualität der Erkennung noch Verbesserungsbedarf.

Die Bewertung wurde an zwölf Teststrukturen anhand eines Testframeworkes demonstriert. Messungen bestätigen, dass die neu entwickelten Strukturen bessere Eigenschaften zur Parameteridentifikation aufweisen als frühere. Weiterhin wurde gezeigt, dass ein Genetischer Algorithmus in der Lage ist, Teststrukturen gezielt nach vorgegebenen Kriterien zu optimieren. Sein Ergebnis hängt stark von der Bewertungsfunktion — in diesem Fall die der Teststrukturen — ab.

Im Folgenden soll ein kurzer Ausblick in Bezug auf die Bewertung von Teststrukturen, den Genetischen Algorithmus und die Modenerkennung mittels Neuro-

nalem Netz erfolgen. Abschließend erfolgen einige Vorschläge zur Darstellung der Ergebnisse einer derart automatisierten Optimierung.

Bei der Bewertung von Teststrukturen ist die Verwendung eines Punktesystems anstelle des Notensystems zu untersuchen, das heißt, eine Struktur bekommt umso mehr Punkte, je besser sie die Bewertungskriterien erfüllt. Damit wäre die Bindung an Notenbereiche aufgehoben, was für die Bewertung, ebenso wie für den Genetischen Algorithmus in Hinsicht auf eine bessere Differenzierung vorteilhaft sein kann. Mit Punkten wäre immernoch eine Intervallbewertung möglich.

Für die Beurteilung der Sensitivität gegenüber Fertigungsparametern sind andere Verfahren zu untersuchen, welche für beliebig viele Parameter anwendbar sind und auch ihre Wechselwirkungen untereinander zu betrachten. Die derzeitige Bewertung ermöglicht keine direkte Trennung danach, welcher Eigenmode sich für die Identifikation eines bestimmten Parameters eignet. Eine Teststruktur wird auch dann als gut bewertet, wenn nur gute Eigenmoden, das heißt mit einer hohen Sensitivität, für einen Parameter existieren.

Weiterhin können die Bewertungen von Eigenfrequenz und Hauptbewegungsrichtung in den Eigenmoden zu einer zusammengefasst werden, da sie, wie sich herausgestellt hat, untrennbar in die Messbarkeit eingehen.

In Bezug auf den Genetischen Algorithmus wäre die Untersuchung anderer Repräsentationen als der Geometrieparameter günstig. Diese Form ist zu unflexibel und nicht ohne Einschränkungen für mehrere Teststrukturen gültig. Die Entwicklung eines Algorithmus zur Erzeugung neuer Grundgeometrien scheint erfolgversprechend.

Der Genetische Algorithmus selbst kann durch spezielle Mutations- und Kreuzungsoperationen, sowie unterschiedliche Einstellungen seiner Parameter optimiert werden. Der Vergleich verschiedener Selektionsschemata bringt unter Umständen eine Verbesserung gegenüber dem standardmäßig eingesetzten. Auch eine Verbesserung der Bewertung und eine Umsetzung in Punktwert nimmt Einfluss auf den Erfolg des Genetischen Algorithmus.

In [CLV07] sind mehrere Ansätze zu mehrdimensionalen Optimierung mittels Evolutionärer Algorithmen aufgeführt. Ihre Eignung in Hinsicht auf die Optimierung von Teststrukturen sollte untersucht werden. Wegen des Aufwandes wurden sie in dieser Arbeit nicht betrachtet beziehungsweise eingesetzt.

Die Eigenmodenerkennung mittels Neuronalem Netz funktioniert prinzipiell. Die Erstellung eines Neuronalen Netzes ist allerdings selbst eine Optimierungsaufgabe. Es ist zu erforschen, ob ein entsprechend angepasstes Netz oder sogar eine andere Netztopologie bessere Ergebnisse liefert.

Für die Wahl der spezifischen Auslenkungen ist des Finden von signifikanten Punkten zu verbessern. Federn und ähnlich feine Strukturen könnten in ANSYS in Komponenten zusammengefasst und mit einer dichteren Auswertung bedacht werden. Weiterhin ist sicherzustellen, dass immer die Auslenkungen an den gleichen Koordinaten verwendet werden. Die optimale Anzahl Eingabedaten und Netzschichten ist ausführlicher zu untersuchen.

Da die Amplitude der Auslenkung in der Simulation auf den maximalen Wert normiert wird, kommt es zu quantitativen Schwankungen der Auslenkungen innerhalb einer Parametervariation. Der Einfluss auf das Neuronale Netz ist festzustellen. Eine Normierung gegen einen konstanten Wert wäre eine Alternative.

Die Zusammenfassung von Ergebnissen in HTML hat den Vorteil, dass sie plattformübergreifend nutzbar ist. Die Generierung der entsprechenden Dokumente ist jedoch aufwendig, da die komplette HTML-Struktur erzeugt werden muss. Günstiger wäre eine spezielle grafische Oberfläche, welche die Daten direkt einlesen und anzeigen kann. Unter Nutzung einer entsprechenden Bibliothek, wie Qt, wäre auch diese Lösung plattformunabhängig. Die Daten könnten auch über XML bereitgestellt und dann entsprechend aufbereitet werden.

In jedem Fall ist eine Übersicht über die einzelnen Teststrukturen anzustreben, die auch grafische Elemente, wie beispielsweise Bilder der Eigenmoden, abbilden kann. Somit bleibt der Optimierungsalgorithmus nachvollziehbar.

Anhang A

Regression und Interpolation von Daten

A.1 Regression

Die Regression der simulierten beziehungsweise gemessenen Werte erfolgt mit Hilfe der Vandermond'schen Matrix. Der Vorteil dieses Verfahrens liegt in der einfachen Anwendung. Die allgemeine Gleichung A.1 erlaubt eine Regression über mehrere Parameter.

$$f(p_1, ..., p_m) = \sum_{\substack{n_i \in \{0, ..., N_{p_i}\} \\ k \in \{0, ..., K\}}} \prod_{i=1}^{m} a_k \cdot p_i^{n_i} \tag{A.1}$$

mit
- p_i ... Parameter i
- N_{p_i} ... Regressionsgrad des Parameters p_i
- k ... Regressionsglied
- a_k ... Koeffizient des Regressionsgliedes k
- m ... Anzahl der Parameter
- K ... Regressionspolynomgrad $K = \prod_{i=1}^{m} (1 + N_{p_i})$

Die Regression erfolgt auf normierten Daten. Auf Basis der Exponentenmatrix, welche in Anhang B näher beschrieben ist, wird die Vandermond'sche Matrix V generiert. Die Koeffizienten der Regressionsfunktion ergeben sich aus Gleichung A.2.

$$V \cdot \vec{a} = \vec{y} \tag{A.2}$$

mit

$$\vec{a} = \begin{pmatrix} a_1 \\ ... \\ a_K \end{pmatrix}$$

als Regressionskoeffizientenvektor, sowie

$$\vec{y} = \begin{pmatrix} y_1 \\ ... \\ y_K \end{pmatrix}$$

als Funktionswertvektor.

Im Fall der Antwortflächen für die Eigenmoden einer Struktur ergibt sich die Vandermond'sche Matrix aus den gemessenen beziehungsweise simulierten Parameterwerten und der Exponentenmatrix wie in Gleichung A.3.

$$V = E \cdot \vec{p} \tag{A.3}$$

Die Funktionswerte sind die jeweiligen Eigenfrequenzen bei einer Eigenmode.

Um den gesuchten Koeffizientenvektor $\vec{a}$ zu erhalten, wird die Gleichung A.2 entsprechend A.4 umgeformt.

$$V^T \cdot V \cdot \vec{a} = V^T \cdot \vec{y} \tag{A.4}$$

Aus $V^T \cdot V$ ergibt sich die reduzierte Matrix V_{red}. Diese ist immer quadratisch. Mit den Gleichungen A.5 und A.6 kann der Koeffizientenvektor $\vec{a}$ aus Gleichung A.7 berechnet werden.

$$V_{\text{red}} = V^T \cdot V \tag{A.5}$$

$$B = V^T \cdot \vec{y} \tag{A.6}$$

$$\begin{aligned} \vec{a} &= V_{\text{red}}^{-1} \cdot B \\ &= (V^T \cdot V)^{-1} \cdot V^T \cdot \vec{y} \end{aligned} \tag{A.7}$$

Für ein vollständig lösbares Gleichungssystem müssen wenigstens K viele Datensätze zur Verfügung stehen.

Bei der Betrachtung mehrerer Eigenmoden werden Koeffizientenvektor $\vec{a}$ und Funktionswertevektor $\vec{y}$ Matrizen (vergleiche Gleichung A.8).

$$\vec{A} = (V^T \cdot V)^{-1} \cdot V^T \cdot \vec{Y} \tag{A.8}$$

Die Dimensionen sind in Tabelle A.1 zusammengestellt. Dabei ist M die Anzahl der Eigenmoden und N die Anzahl der Datensätze ($N \geq K$).

Tabelle A.1: Dimensionen der Matrizen

Matrix	# Spalten	# Zeilen
V	N	K
V_{red}	K	K
Y	M	K
A	M	K

A.2 Interpolation

Bei der Interpolation wird aus gegebenen Parameterwerten, der Exponentenmatrix und den Koeffizienten der Regression der zugehörige Funktionswert wie in Gleichung A.9 ermittelt.

$$y = E \cdot \vec{p}^{T} \cdot \vec{a}_m \tag{A.9}$$

mit y ... Funktionswert (im speziellen Fall Eigenfrequenz)
E ... Exponentenmatrix
$\vec{p}$ ... Vektor der Parameter $(p_1, ..., p_i)$
$\vec{a}_m$ ... Koeffizientenvektor für eine Eigenmode m.

Der relative Fehler e der Regression für ein Parameterset lässt sich aus dem Sollwert der Eigenfrequenz f_{soll} und der aus der Interpolation bestimmten f_{interpol} berechnen (siehe Gleichung A.10).

$$e = \left| \frac{f_{\text{interpol}} - f_{\text{soll}}}{f_{\text{soll}}} \right| \tag{A.10}$$

Über die gesamten Werte kann eine statistische Auswertung für die Regression einer Antwortfläche für jeweils eine Eigenmode durchgeführt werden.

Anhang B

Erstellung der Vandermond'schen Matrix

Die Vandermond'sche Matrix wird für die Polynominterpolation nach Gleichung A.1 eingesetzt. Der Ansatz für zwei Parameter ist in Gleichung B.1 gegeben.

$$f(p_1, p_2) = a_0 + a_1 p_1^0 p_2^0 + a_2 p_1^1 p_2^0 + \cdots + a_K p_1^n p_2^m \tag{B.1}$$

mit n ... Interpolationsgrad des Parameters p_1
m ... Interpolationsgrad des Parameters p_2
a_i ... Polynomkoeffizienten des Polynoms i
K ... Anzahl der Regressionsglieder.

Die Vandermond'sche Matrix hat für zwei Parameter mit dem jeweiligen Interpolationsgrad 1 folgendes Aussehen:

$$V = \begin{bmatrix} 1 & p_1^0 & p_2^0 \\ 1 & p_1^1 & p_2^0 \\ 1 & p_1^0 & p_2^1 \\ 1 & p_1^1 & p_2^1 \end{bmatrix} \tag{B.2}$$

Das Interpolationspolynom ist $f(p_1, p_2) = a_0 + a_1 p_1 + a_2 p_2 + a_3 p_1 p_2$.

Die Anzahl der Koeffizienten K lässt sich durch Gleichung B.3 errechnen. Damit ergeben sich für den beschriebenen Fall vier Koeffizienten.

$$K = \prod_{i=0}^{n} (1 + p(x_i)) \tag{B.3}$$

mit p ... Polynomgrade des Parameters x_i
n ... Anzahl der Parameter.

Um eine Vandermond'sche Matrix dynamisch aufzubauen, ist ein algorithmisches Vorgehen notwendig. Eine Möglichkeit wären verschachtelte Schleifen, wobei dann die Anzahl der Parameter statisch ist. Dieser Nachteil läßt sich mit dem Operieren auf Feldern beheben. Dabei werden die Grade der Exponenten für jeden Parameter bei jedem Polynomkoeffizienten berechnet und in einer sogenannten Exponentenmatrix hinterlegt.

Ausgangspunkt ist ein Feld mit n Zellen, die den Parametern entsprechen (siehe Abbildung B.1). In den Zellen stehen die Interpolationsgrade n_{p_i} der Parameter p_i.

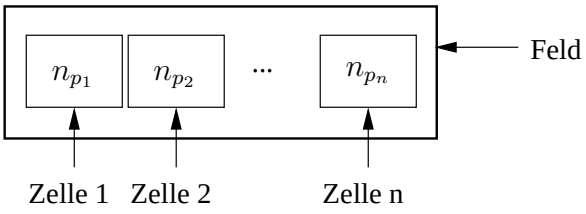


Abbildung B.1: Feld der Interpolationsgrade n_{p_i} der Parameter p_i

Die Anzahl der Koeffizienten und damit der Gesamtschritte des Algorithmus lässt sich über Gleichung B.3 ermitteln.

Jede Kombination der Interpolationsgrade wird mit einem Koeffizienten multipliziert. Das Gleichungssystem ist in Gleichung B.4 dargestellt.

$$E \cdot \vec{p} \cdot \vec{a} = \vec{f} \tag{B.4}$$

mit E ... Exponentenmatrix

$\vec{p}$ ... Matrix der Parameter $\begin{bmatrix} p_1(F_1) & \dots & p_i(F_1) \\ p_1(F_2) & \dots & p_i(F_2) \\ \vdots & & \vdots \\ p_1(F_j) & \dots & p_i(F_j) \end{bmatrix}$

$\vec{a}$ ... Koeffizientenvektor

$\vec{f}$ ... Vektor der Funktionswerte.

Die Exponentenmatrix besteht aus den Kombinationen der Interpolationsgrade. Dafür wird für jede Zeile jeweils bei einem Parameter der Interpolationsgrad von 0 aus inkrementiert, bis er den Wert in der entsprechenden Zelle des Feldes erreicht. Ist dieser erreicht, erfolgt ein Übertrag auf die nächsthöhere Zelle, bei der der Grad eines Parameters nicht Null ist. Alle niedrigeren Zellen werden wieder auf 0 gesetzt, der Vorgang beginnt von neuem wie im Programmablaufplan Abbildung B.2 und dem Unterprogramm B.3 dargestellt, bis alle Polynomgliederexponenten ermittelt sind.

Dieser Vorgang einspricht einer vollständigen Enumeration wie bei einem Stellenwertsystem. Die möglichen Ziffern einer Stelle sind dabei in dem Feld der Interpolationsgrade festgelegt.

Damit ergibt sich für drei Parameter mit den Graden 2, 1 und 1 eine Gesamtkoeffizientenzahl von $K = 3 \cdot 2 \cdot 2 = 12$. Das Parameterfeld ist

p_1	p_2	p_3
2	1	1

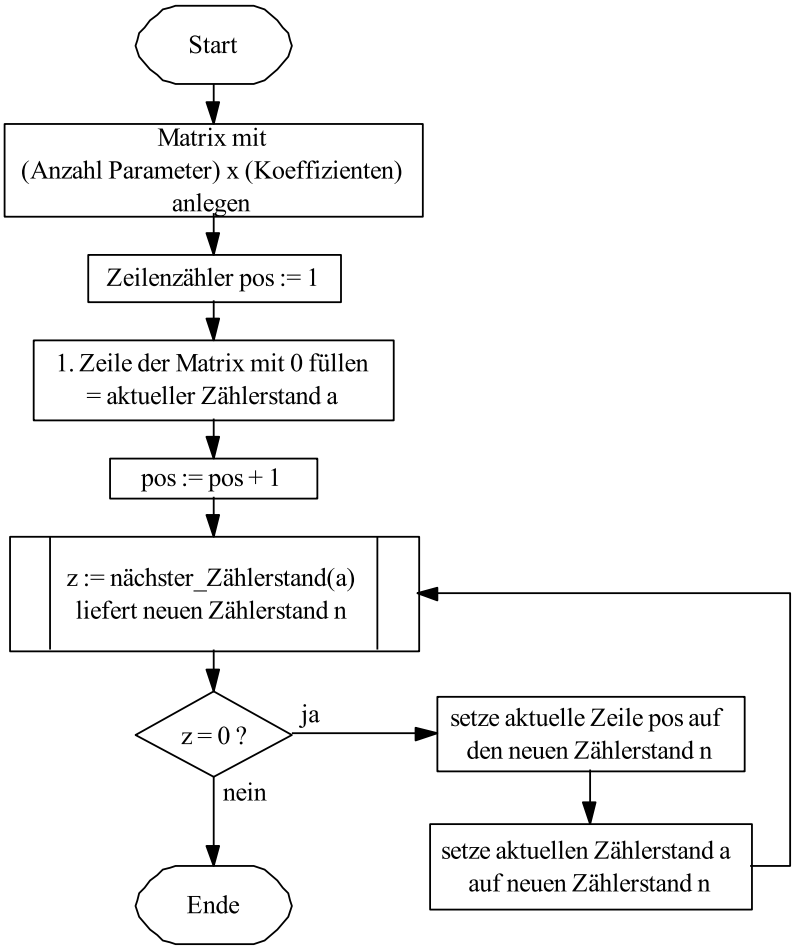


Abbildung B.2: Hauptroutine zum Ermitteln der Exponentenmatrix

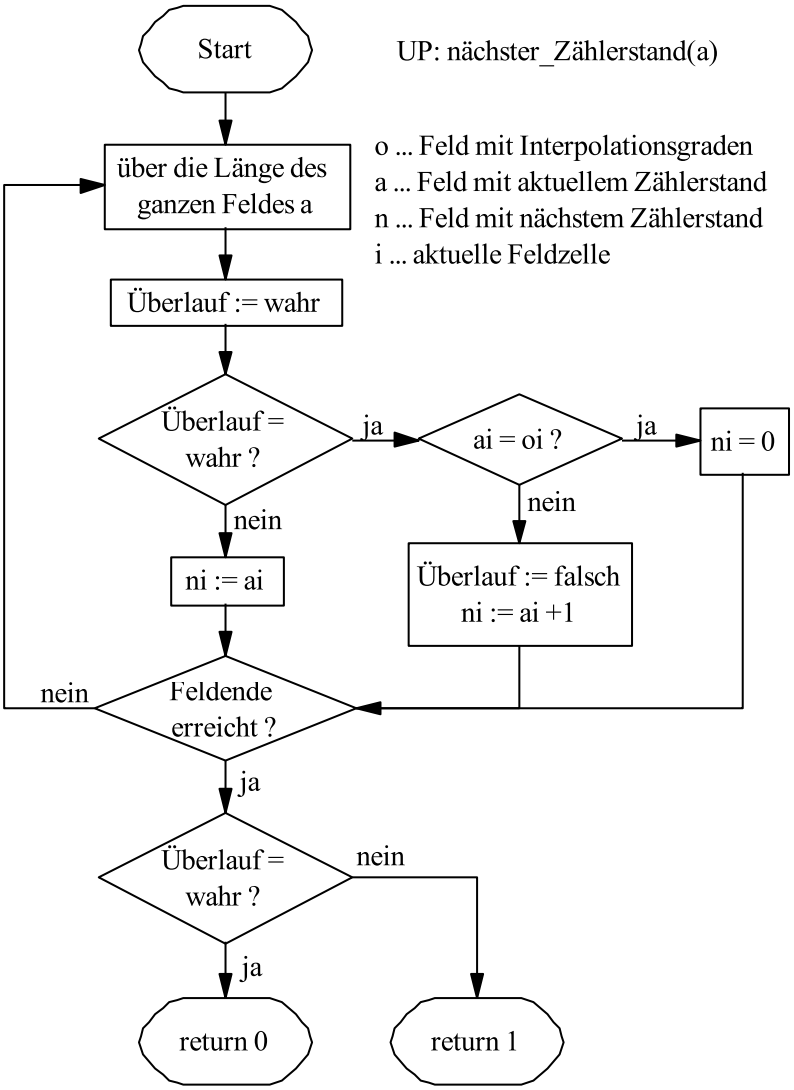


Abbildung B.3: Unterroutine zum Ermitteln der Exponentenmatrix

und die Exponentenmatrix damit

$$E^T = \begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Die Ableitung der resultierenden Funktion nach einem Parameter ist über eine Exponentenmatrix zu realisieren. Der Vorgang erfolgt zeilenweise. Der neue Koeffizient einer Zeile ergibt sich aus dem alten, mit dem entsprechenden Exponenten multipliziert. In der Exponentenmatrix wird von diesem Exponenten 1 subtrahiert, wenn dieser nicht 0 ist.

Für die Vandermond'sche Matrix wird die Exponentenmatrix mit dem Parametervektor multipliziert (siehe Gleichung B.5).

$$V = E \cdot \vec{p}^T = E \cdot \begin{pmatrix} p_1 \\ \vdots \\ p_i \end{pmatrix} \tag{B.5}$$

Anhang C

Vertauschen von Eigenmoden

Dieses Kapitel soll das Vertauschen von Eigenmoden bei einer Parametervariation demonstrieren. Dieser Effekt tritt sowohl bei der Simulation als auch bei der Messung auf. Die Ursache liegt in den Änderungen des Verhältnisses der Strukturabmaße.

Abbildung C.1 zeigt die Abhängigkeit der Eigenfrequenzen von der Federbreite für drei Eigenmoden. Es wird deutlich, dass bei einer Federbreite von etwa $6,5\,\mu m$ die Eigenfrequenz des Torsionsmodes die des Out-of-Plane-Modes übersteigt. Bei einer Sortierung nach aufsteigenden Frequenzen würde dies einen Eigenmodentausch zur Folge haben.

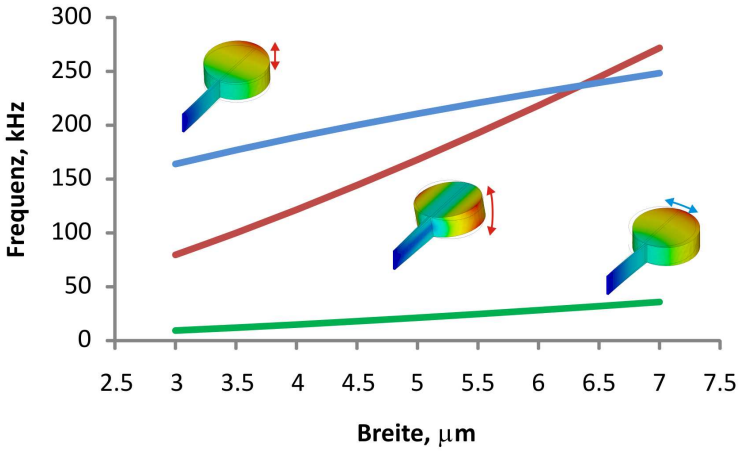


Abbildung C.1: Abhängigkeit der Eigenfrequenzen von der Federbreite

Abbildung C.2 stellt die gemessenen Eigenmoden mit aufsteigender Frequenz bei einer Federbreite von $5\,\mu m$ dar. In Abbildung C.3 ist die Federbreite größer, die Eigenmoden sind vertauscht.

Bei kleinen Abweichungen vom Normparameterwert tritt der Eigenmodentausch nicht in Erscheinung. Die Grenze ist für jede Teststruktur gesondert festzulegen.

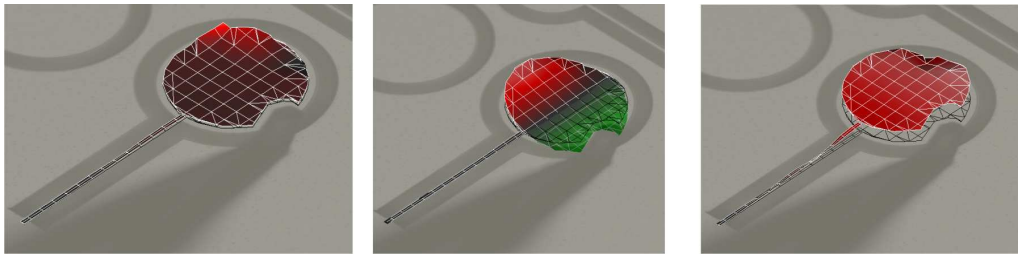


Abbildung C.2: Messung - Eigenmodenreihenfolge (Federbreite=5 μm)

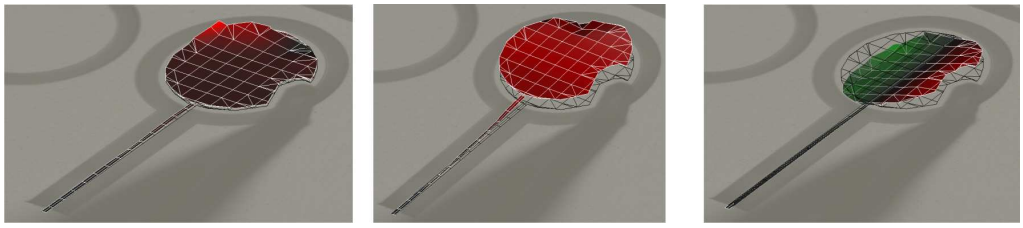


Abbildung C.3: Messung - Eigenmodenreihenfolge (Federbreite=7 μm)

Wird die Reihenfolge der Eigenmoden nicht an die bei Normparametern angpasst, finden sich Spitzen in den Daten, aus denen die Antwortfläche generiert wird und bei Regression dieser steigt der Regressionsfehler. Die ermittelte Antwortfläche entspricht somit nicht der Realität.

Anhang D

Teststrukturgeometrien

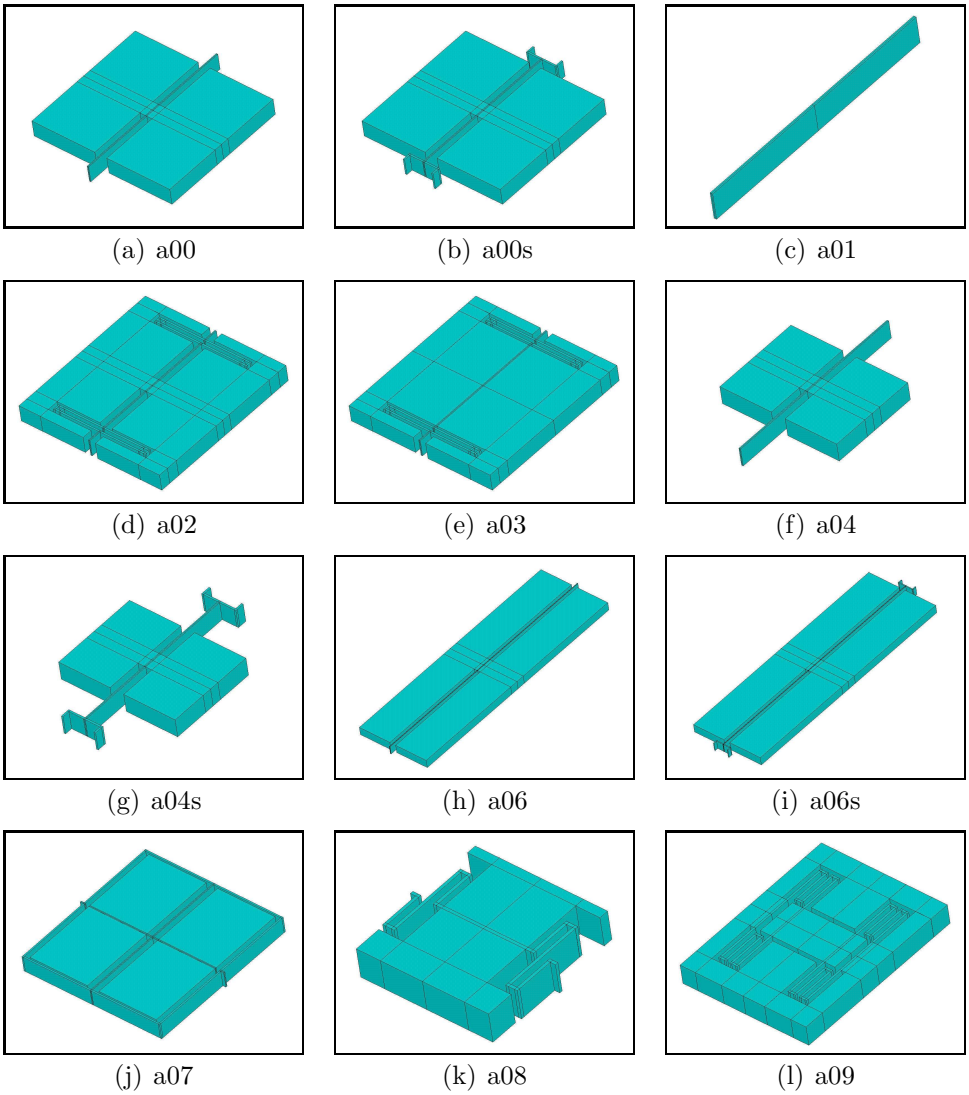
In dieser Arbeit wurden mehrere Grundgeometrien für Teststrukturen entwickelt und untersucht. Diese sind nicht optimiert, bieten aber eine Grundlage dafür. In Abbildung D.1 sind alle Geometrien dargestellt. Mit Ausnahme der Struktur `a08` sind alle aufgeführten viertelsymmetrisch. In diesem Kapitel sind die Eckdaten und die ersten zehn Eigenfrequenzen jeder Teststruktur zusammengefaßt.

Tabelle D.1 führt jeweils die Anzahl der Eigenfrequenzen auf, welche im messbaren Bereich, das heißt unterhalb $500\,kHz$ liegen. Diese werden zusätzlich in In-Plane und Out-of-Plane unterschieden. Weiterhin in die lateralen Ausdehnungen angegeben. Die Strukturdicke entspricht dem Normwert von $50\,\mu m$. Die Noten der Teststrukturen ergeben sich aus der Bewertung.

Tabelle D.1: Teststrukturdaten

Teststruktur	Eigenfrequenzen $< 500\,kHz$	davon Out-of-Plane	Abmessungen x/ μm	y/ μm	Note
a00	6	3	460	500	2
a00s	6	3	460	500	2
a01	1	0	460	5	5
a02	10	5	460	500	2
a03	8	3	460	500	2,5
a04	5	3	340	500	2
a04s	6	3	340	500	2
a06	10	5	460	1500	2
a06s	10	5	460	1500	2
a07	10	4	500	500	2
a08	6	3	230	480	3
a09	10	3	338	440	2,5

Die Bewertung erfolgte nach den in Abschnitt 4.1 festgelegten Kriterien. Die betrachteten Parameter beschränken sich auf die Federbreite und die Keiligkeit. Es wird deutlich, dass die Struktur `a01` nicht zur Parameteridenifikation geeignet ist,



(a) a00 (b) a00s (c) a01

(d) a02 (e) a03 (f) a04

(g) a04s (h) a06 (i) a06s

(j) a07 (k) a08 (l) a09

Abbildung D.1: Teststrukturgeometrien

da ihre Eigenfrequenzen zu hoch liegen. Dies spiegelt sich auch in der Bewertung wider.

Die restlichen Teststrukturen sind prinzipiell gut geeignet. Unterschiede liegen in der Anzahl der verwertbaren Eigenfrequenzen und den Parametersensitivitäten. Die Ähnlichkeit der Noten wird durch die geringe Differenzierung der Bewertung verursacht. Durch Anpassung der Bewertungskriterien und -intervalle kann eine feinere Auswahl geeigneter Teststrukturen für eine Aufgabe getroffen werden.

Abbildung D.2 und D.3 zeigen die ersten zehn Eigenfrequenzen der Teststrukturen für die Normwerte der Fertigungsparameter Federbreite $x = 5\,\mu m$, Strukturdicke $th = 50\,\mu m$ und Keiligkeit $x = 0,2\,\mu m$.

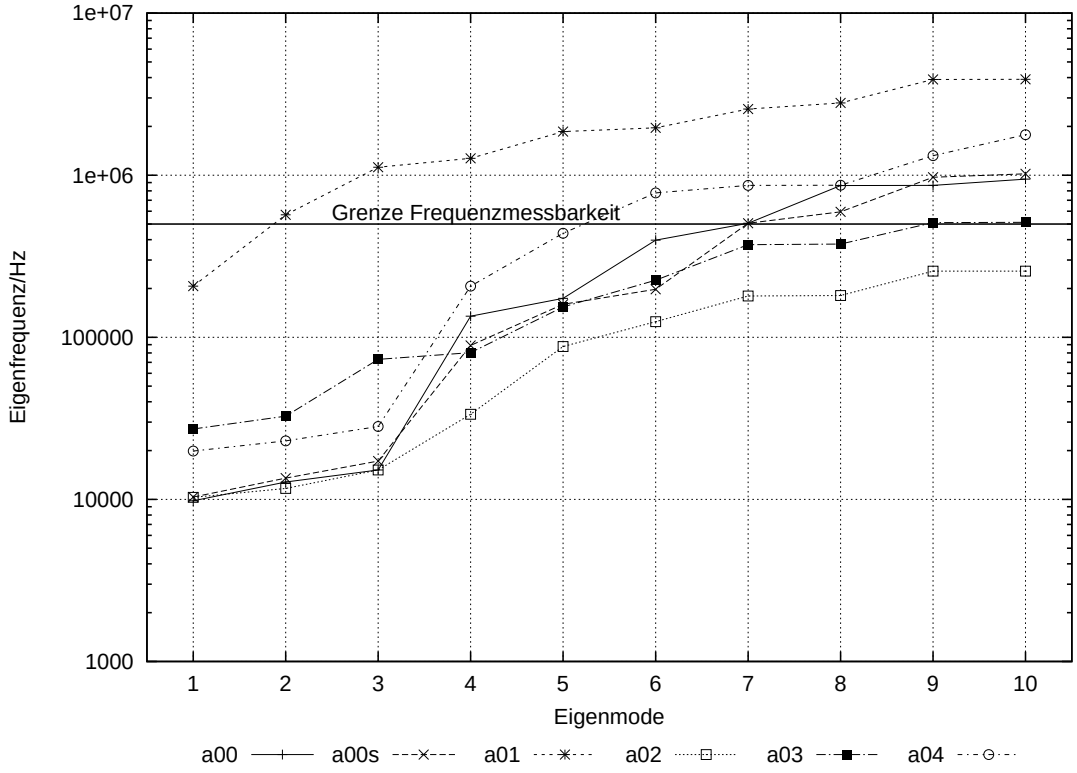


Abbildung D.2: Eigenfrequenzen der Teststrukturen - Teil 1

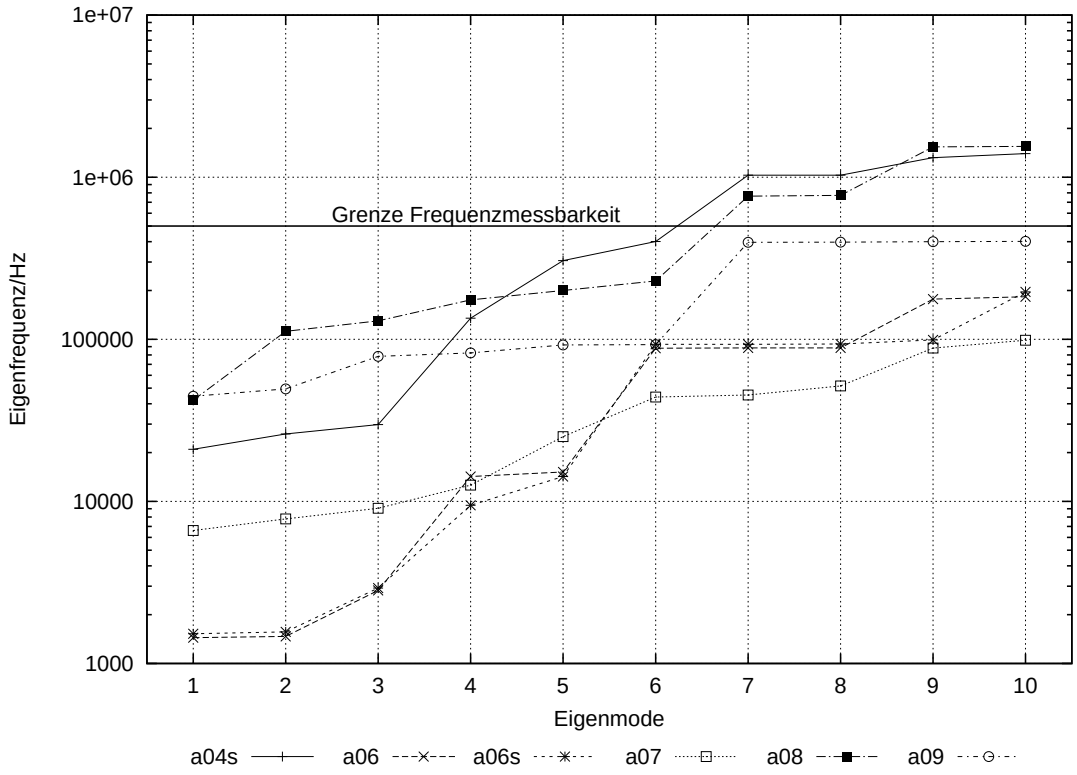


Abbildung D.3: Eigenfrequenzen der Teststrukturen - Teil 2

Anhang E

Daten zur Eigenmodenerkennung

In diesem Kapitel werden zusätzliche Daten zur Eigenmodenerkennung mittels Neuronalem Netz bereitgestellt. Die Abbildungen E.1 bis E.5 zeigen die Häufigkeiten erkannter Parametersets bei verschiedenen Testszenarien (siehe Abschnitt 5.1.1). Jede Testreihe enthielt 24 Parametersets, mit welchen jeweils hundert Erkennungen durchgeführt wurden. Es wird deutlich, das die Erkennung mit einer geringeren Anzahl Eigenmoden besser abschneidet. Die Ursachen werden in Abschnitt 5.1.1 diskutiert. Die Parameter dieser simulierten Struktur weisen höhere Schwankungsbreite als die eigentlichen Teststrukturen auf, um die Auswirkungen des Modentauschs zu demonstrieren.

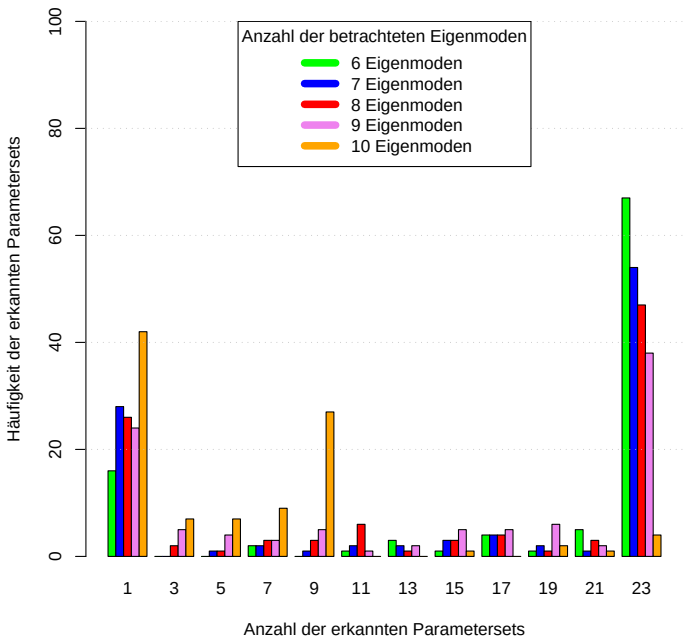


Abbildung E.1: Häufigkeitsverteilung erkannter Parametersets bei udispl1

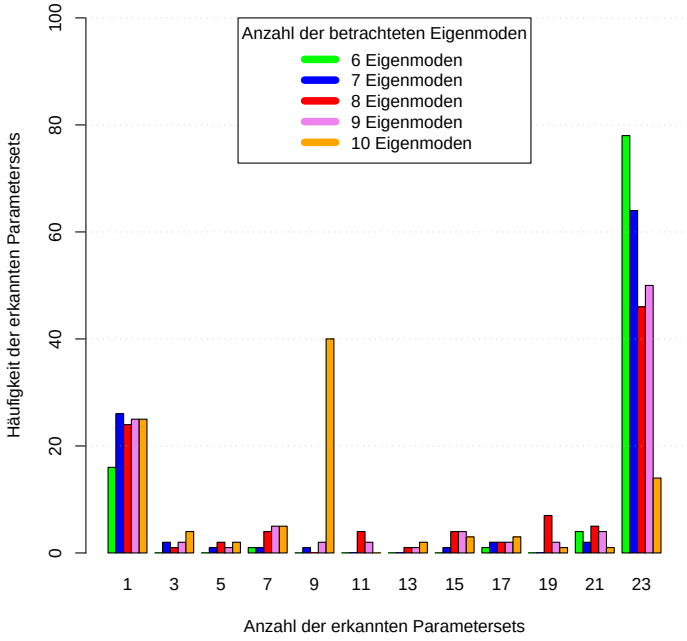


Abbildung E.2: Häufigkeitsverteilung erkannter Parametersets bei udispl2

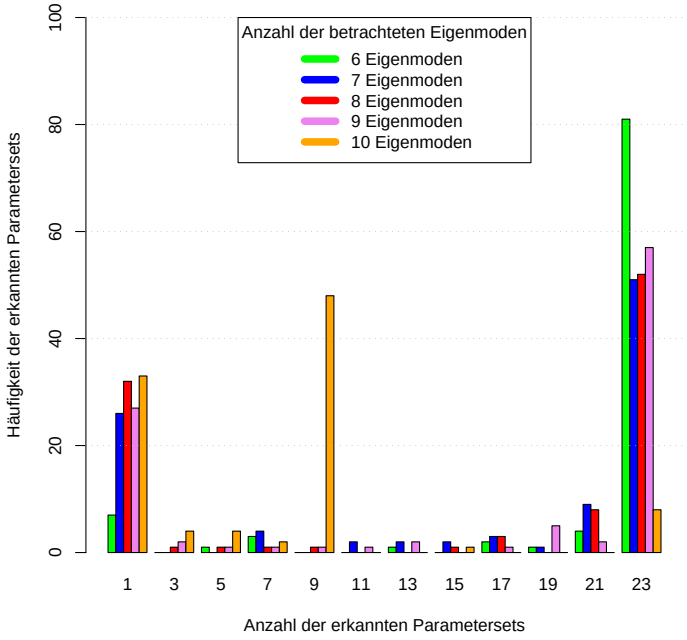


Abbildung E.3: Häufigkeitsverteilung erkannter Parametersets bei udispl3

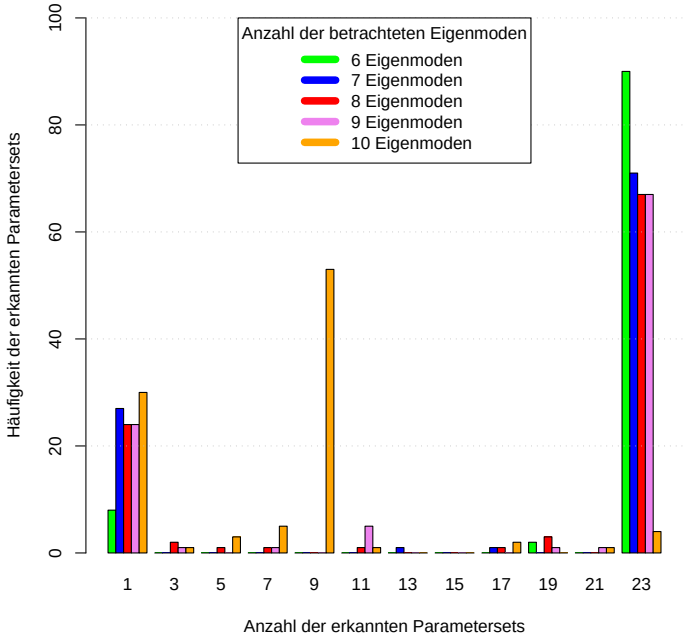


Abbildung E.4: Häufigkeitsverteilung erkannter Parametersets bei udispl4

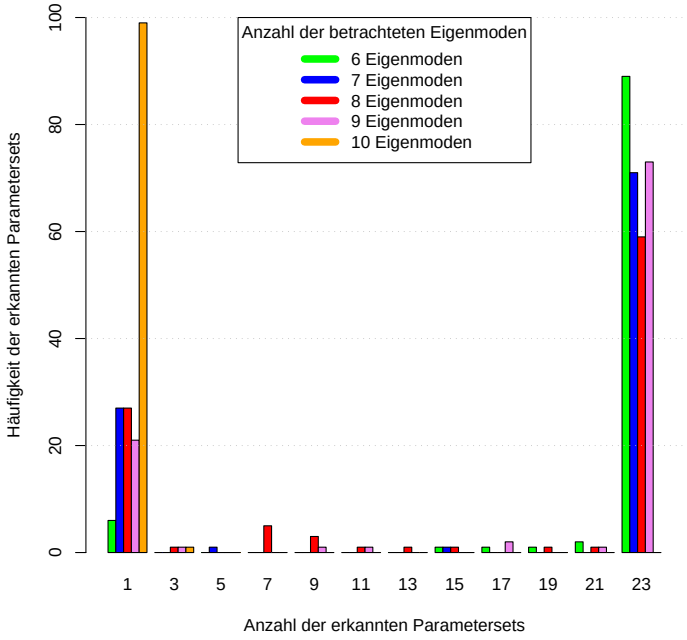


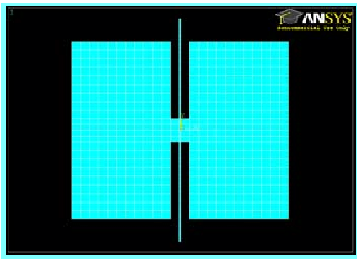
Abbildung E.5: Häufigkeitsverteilung erkannter Parametersets bei udispl5

Anhang F

Beispielbewertung in MathCAD

Beispielbewertung anhand der Parameter Dicke (x) und Keiligkeit (y)

Teststruktur a00



"../data/a00/file000.jpg"

Dieses Skript verarbeitet Simulationsdaten einer Teststruktur zur einer Bewertung von dieser.

Datenquelle ist eine Parametervariation mit sortierten Eigenmoden durch ein neuronales Netz.

Schrittfolge:

- **Einstellen der Teststrukturdaten**
- **Erstellen der Antwortflächen aus den Daten**
- **Bewertung der Teststruktur anhand bestimmter Bewertungskritierien**

Die blau hervorgehobenen Werte sind für die Teststruktur entsprechend anzupassen.

Einstellungen zur Teststruktur

Quelldaten

src := "..\data\"

ts_name := "a00"

modefile := verkett(src, ts_name, "\", ts_name, "_", "modeform.dat")

Normwerte der Parameter

norm_p1 := 50 Strukturdicke

norm_p2 := 0.1 Keiligkeit

Daten der Parametervariation

p1_min := 48	p1_max := 52	p1_step := 0.5	Strukturdicke
p2_min := 0.1	p2_max := 0.3	p2_step := 0.1	Keiligkeit

Simulations- und Berechnungsparameter

Anzahl der Eigenmoden

N_mode := 10

Anzahl der Parameter

N_par := 2

Achtung: Bei Änderung der Anzahl der Parameter müssen die Regression und die daraus resultierenden Funktionen angepasst werden.

Grad der Regressionsfunktion

n := 1

Indizierung und Rechentoleranzen

ORIGIN := 1 TOL := 10^{-7}

Daten der Parametervariation (sortierte Eigenmoden)

data := ...\a00_loop.dat

Modeformen

modes := PRNLESEN(modefile)

data := submatrix(data, 1, zeilen(data), 2, spalten(data))

modes := submatrix(modes, 1, 1, 4, spalten(modes))

modes =

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	1	0	1	1	1

Verarbeitung der Daten

Ermitteln der Anzahl der Datensätze

N := letzte(data$^{\langle 1 \rangle}$) i := 0 .. N

Matrix der Parameter

M := submatrix(data, 1, N, 1, N_par)

Darstellungseinstellung

maximale Frequenz der Messwerte:

f_mess_max := max(submatrix(data, 1, N, N_par + 1, N_par + N_mode))

f_mess_max = 9.448×10^{5}

maximale Frequenz in Diagrammen:

f_dia_max := 10^{6}

Regression

Exponenten := $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

$$\text{VMM} := \text{for } \text{datasets} \in 1 \,..\, \text{zeilen(data)}$$
$$\quad \text{produkt} \leftarrow 1$$
$$\quad \text{for } i \in 1 \,..\, \text{spalten(Exponenten)}$$
$$\quad\quad \text{for } j \in 1 \,..\, \text{zeilen(Exponenten)}$$
$$\quad\quad\quad \text{parameterwert} \leftarrow M_{\text{datasets},j}$$
$$\quad\quad\quad \text{help} \leftarrow \begin{cases} 1 & \text{if } \text{parameterwert} = 0 \wedge \text{Exponenten}_{j,i} = 0 \\ \text{parameterwert}^{\text{Exponenten}_{j,i}} & \text{otherwise} \end{cases}$$
$$\quad\quad\quad \text{produkt} \leftarrow \begin{cases} \text{help} & \text{if } j = 1 \\ \text{produkt} \cdot \text{help} & \text{otherwise} \end{cases}$$
$$\quad\quad\quad \text{produkt}$$
$$\quad\quad \text{VMM}_{\text{datasets},i} \leftarrow \text{produkt}$$

$$\text{VMMred} := \text{VMM}^{T} \cdot \text{VMM}$$

$$\text{koeffs} := \text{for } i \in 1 \,..\, \text{N\_mode}$$
$$\quad B \leftarrow \text{VMM}^{T} \cdot \text{data}^{\langle \text{N\_par}+i \rangle}$$
$$\quad \text{koeffs}^{\langle i \rangle} \leftarrow \text{VMMred}^{-1} \cdot B$$

$$f(x,y,i) := \text{koeffs}_{1,i} + \text{koeffs}_{2,i} \cdot x + \text{koeffs}_{3,i} \cdot y + \text{koeffs}_{4,i} \cdot x \cdot y$$

$\text{koeffs} =$

	1	2	3	4	5	6	7
1	$9.178 \cdot 10^3$	$1.292 \cdot 10^4$	$1.529 \cdot 10^4$	$1.332 \cdot 10^4$	$1.98 \cdot 10^4$	$3.952 \cdot 10^5$	$1.368 \cdot 10^5$
2	18.156	5.197	6.773	$2.456 \cdot 10^3$	$3.114 \cdot 10^3$	96.053	$7.394 \cdot 10^3$
3	$-2.432 \cdot 10^3$	$-3.675 \cdot 10^3$	$-4.321 \cdot 10^3$	-337.777	$-1.367 \cdot 10^3$	$-1.606 \cdot 10^4$	$3.938 \cdot 10^3$
4	-7.012	-0.348	-0.822	-237.945	-278.981	-10.181	-204.629

Fehler der Regression

maximaler Regressionsfehler für jeden Eigenmode

$$\text{Err} := \text{i} \leftarrow 1$$
$$\quad \text{for } i \in 1 \,..\, \text{zeilen(data)}$$
$$\quad\quad \text{for } j \in 1 \,..\, \text{N\_mode}$$
$$\quad\quad\quad \text{Err}_{i,j} \leftarrow \left| \frac{f\left(\text{data}_{i,1}, \text{data}_{i,2}, j\right) - \text{data}_{i,j+\text{N\_par}}}{\text{data}_{i,j+\text{N\_par}}} \right|$$
$$\quad \text{for } j \in 1 \,..\, \text{N\_mode}$$
$$\quad\quad \text{tmp1}_j \leftarrow \max\left(\text{Err}^{\langle j \rangle}\right)$$
$$\quad \text{tmp1}$$

$\text{Err}^{T} =$

	1	2	3	4	5	6	7
1	0.022	0.027	0.027	0.02	0.019	$2.744 \cdot 10^{-3}$	0.022

%

Abhängigkeiten der Parameter von den Eigenfrequenzen

Parameter 1 (Dicke)

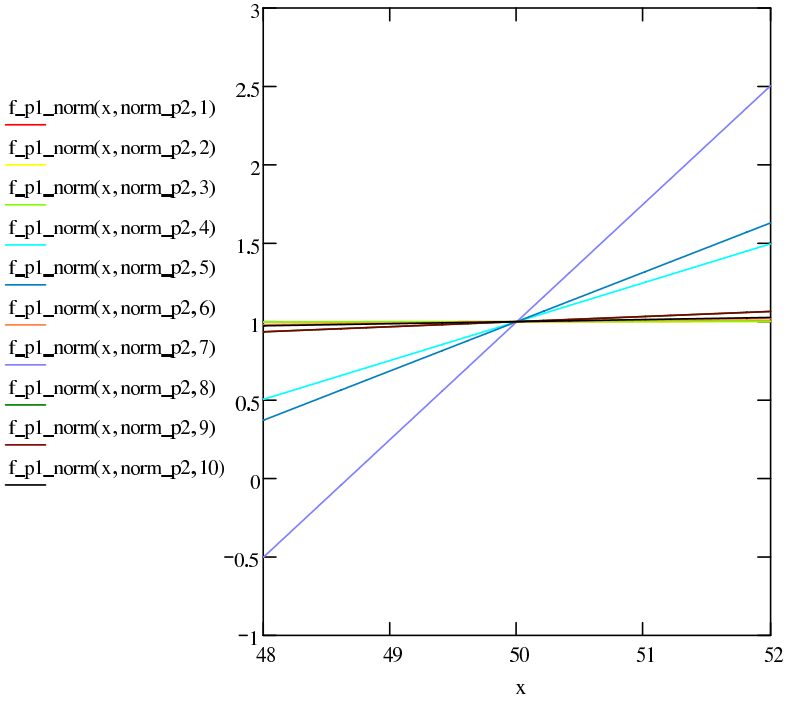
Normierung nach der ersten Eigenfrequenz

Vektor der Abstände der jeweiligen Funktion zur Normgerade des Eigenmodes

$$\text{norm\_x} := \text{for } i \in 1 .. \text{N\_mode}$$
$$\text{norm\_x}_i \leftarrow f(\text{norm\_p1}, \text{norm\_p2}, i) - f(\text{norm\_p1}, \text{norm\_p2}, 1)$$

Normierte Funktion des i-ten Eigenmodes in Abhängigkeit von Parameter 1

$$\text{f\_p1\_norm}(x, y, i) := \frac{f(x, y, i) - \text{norm\_x}_i}{f(\text{norm\_p1}, \text{norm\_p2}, 1)}$$



Parameter 2 (Keiligkeit)

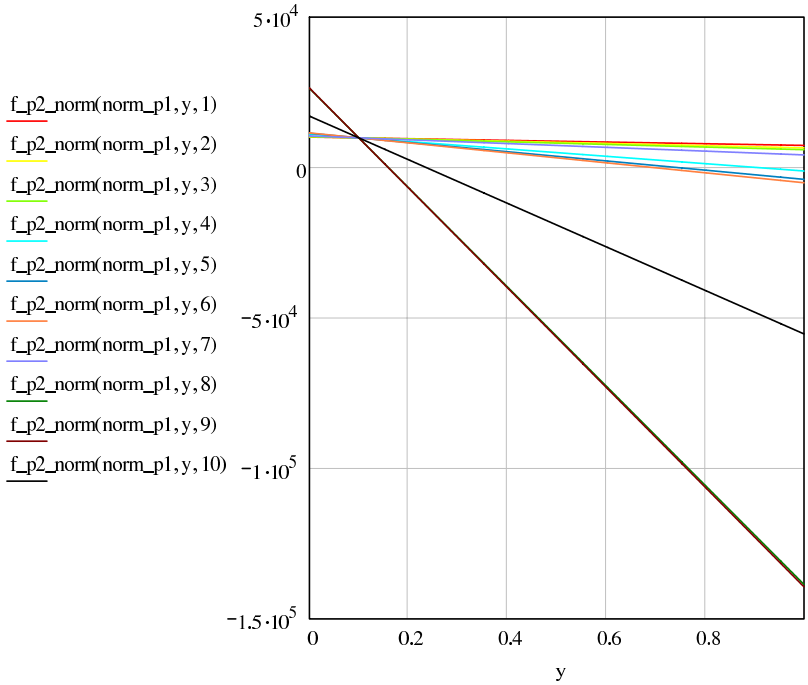
Normierung nach der ersten Eigenfrequenz

Vektor der Abstände der jeweiligen Funktion zur Normgerade des Eigenmodes

$$\text{norm\_y} := \text{for } i \in 1 .. \text{N\_mode}$$
$$\text{norm\_y}_i \leftarrow f(\text{norm\_p1}, \text{norm\_p2}, i) - f(\text{norm\_p1}, \text{norm\_p2}, 1)$$

Normierte Funktion des i-ten Eigenmodes in Abhängigkeit von Parameter 1

$$\text{f\_p2\_norm}(x, y, i) := f(x, y, i) - \text{norm\_y}_i$$



Anstiege der Abhängigkeitsgeraden der Eigenmoden

Parameter 1

Ableitung:

$df\_p1(i) := koeffs_{2,i}$

Koeffizientenvektor der Eigenmoden:

$grad\_p1 :=$ for $i \in 1..10$
$grad\_x_i \leftarrow df\_p1(i)$

	1
1	18.156
2	5.197
3	6.773
4	2.456·103
5	3.114·103
6	96.053
7	7.394·103
8	327.985
9	327.79
10	131.232

grad_p1 =

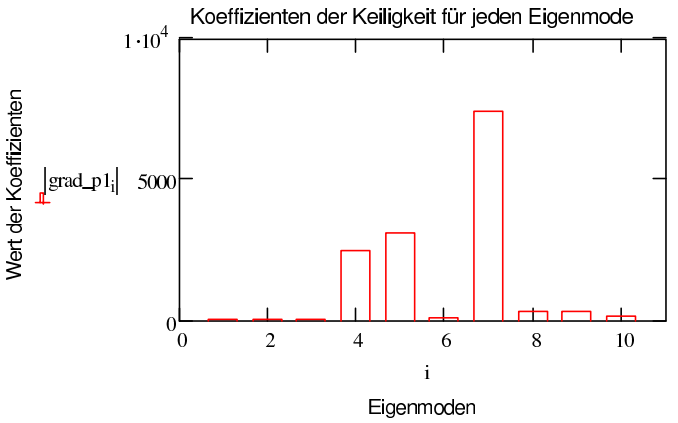
Parameter 2

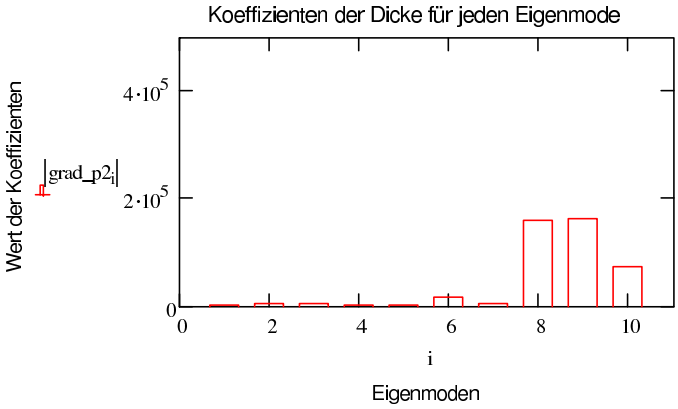
$df\_p2(i) := koeffs_{3,i}$

$grad\_p2 :=$ for $i \in 1..10$
$grad\_y_i \leftarrow df\_p2(i)$

	1
1	-2.432·103
2	-3.675·103
3	-4.321·103
4	-337.777
5	-1.367·103
6	-1.606·104
7	3.938·103
8	-1.6·105
9	-1.61·105
10	-7.196·104

grad_p2 =





Bewertung der Teststruktur:

Die Eigenmoden der Teststruktur werden anhand ausgewählter Bewertungsparameter bewertet.
Ihre Bewertung wird zu einer Gesamtbewertung der Teststruktur verknüpft.

Bewertung der Frequenz:

$$freqs := submatrix(data, 1, 1, N\_par + 1, N\_par + N\_mode)$$

```
eval_freq := | for j ∈ 1 .. N_mode
             |   eval ← 5  if freqs_{1,j} > 500·10^3
             |   eval ← 4  if 100·10^3 < freqs_{1,j} ≤ 500·10^3
             |   eval ← 2  if 10·10^3 < freqs_{1,j} ≤ 100·10^3
             |   eval ← 1  if freqs_{1,j} ≤ 10·10^3
             |   eval_freq_{1,j} ← eval
             |   eval_freq
             | eval_freq
```

evaluation_frequency := stapeln(freqs, eval_freq)

evaluation_frequency =

	1	2	3	4	5	6
1	9.773·10^3	1.28·10^4	1.518·10^4	1.3·10^5	1.678·10^5	3.981·10^5
2	1	2	2	4	4	4

Noten der Frequenz

eval_freq =

	1	2	3	4	5	6	7	8	9	10
1	1	2	2	4	4	4	4	5	5	5

Bewertung der Moden:

```
eval_modes := | for j ∈ 1 .. N_mode
              |   eval ← 1  if modes_{1,j} = 0
              |   eval ← 2  if modes_{1,j} = 1
              |   eval_modes_{1,j} ← eval
              |   eval_modes
              | eval_modes
```

evaluation_modes := stapeln(modes, eval_modes)

evaluation_modes =

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	1	0	1	1	1
2	1	2	2	1	1	2	1	2	2	2

Noten der Modeformen

eval_modes =

	1	2	3	4	5	6	7	8	9	10
1	1	2	2	1	1	2	1	2	2	2

Bewertung der Koeffizienten:

Parameterauflösung: $dp := 50 \cdot 10^{-3}$

relative Frequenzschwankung: $dfres(m) := |m \cdot dp|$

Frequenzschwankung über gesamten Parameterbereich: $dfges\_p1(m) := |m \cdot (p1\_min - p1\_max)|$

$dfges\_p2(m) := |m \cdot (p2\_min - p2\_max)|$

Bewertung der Sensitivität nach der Keiligkeit x:

$$koeff\_p1 := grad\_p1^{T}$$

$$\begin{aligned}
eval_p1 := \;& \text{for } j \in 1..N_mode \\
& \quad eval \leftarrow 1 \text{ if } dfres\left(koeff_p1_{1,j}\right) > 1\cdot 10^{3} \\
& \quad eval \leftarrow 2 \text{ if } 20 < dfres\left(koeff_p1_{1,j}\right) \le 1\cdot 10^{3} \\
& \quad eval \leftarrow 3 \text{ if } 1 < dfres\left(koeff_p1_{1,j}\right) \le 20 \\
& \quad eval \leftarrow 5 \text{ if } dfres\left(koeff_p1_{1,j}\right) \le 1 \\
& \quad eval \leftarrow 5 \text{ if } dfges_p1\left(koeff_p1_{1,j}\right) \le 0.001\cdot f(norm_p1, norm_p2, j) \\
& \quad eval_p1_{1,j} \leftarrow eval \\
& \quad eval_p1 \\
& eval_p1
\end{aligned}$$

$$evaluation\_p1 := stapeln(koeff\_p1, eval\_p1)$$

$evaluation\_p1 =$

	1	2	3	4	5	6
1	18.156	5.197	6.773	2.456·10³	3.114·10³	96.053
2	5	5	5	2	2	5

Noten der Sensitivität nach x

$eval\_p1 =$

	1	2	3	4	5	6	7	8	9	10
1	5	5	5	2	2	5	2	3	3	5

Bewertung der Sensitivität nach der Dicke th:

$$koeff\_p2 := grad\_p2^{T}$$

$$\begin{aligned}
eval_p2 := \;& \text{for } j \in 1..N_mode \\
& \quad eval \leftarrow 1 \text{ if } dfres\left(koeff_p2_{1,j}\right) > 1\cdot 10^{3} \\
& \quad eval \leftarrow 2 \text{ if } 20 < dfres\left(koeff_p2_{1,j}\right) \le 1\cdot 10^{3} \\
& \quad eval \leftarrow 3 \text{ if } 1 < dfres\left(koeff_p2_{1,j}\right) \le 20 \\
& \quad eval \leftarrow 5 \text{ if } dfres\left(koeff_p2_{1,j}\right) \le 1 \\
& \quad eval \leftarrow 5 \text{ if } dfges_p2\left(koeff_p2_{1,j}\right) \le 0.001\cdot f(norm_p1, norm_p2, j) \\
& \quad eval_p2_{1,j} \leftarrow eval \\
& \quad eval_p2 \\
& eval_p2
\end{aligned}$$

$$evaluation\_p2 := stapeln(koeff\_p2, eval\_p2)$$

Für eine gute Eignung für mehrere Parameter sollten deren Noten in gleichen Moden möglichst weit auseinander liegen.

```
eval_suit := | for j ∈ 1..N_mode
             |   eval ← 5 − |eval_p1[1,j] − eval_p2[1,j]|
             |   eval_suit[1,j] ← | rund(eval)  if eval ≤ 5
             |                    | 5  otherwise
             |   eval_suit
             | eval_suit
```

evaluation_suitability := stapeln(eval_p1, eval_p2, eval_suit)

evaluation_suitability =

	1	2	3	4	5	6	7	8	9	10
1	5	5	5	2	2	5	2	3	3	5
2	2	2	2	5	2	2	2	1	1	1
3	2	2	2	2	5	2	5	3	3	1

Noten der Eignung

eval_suit =

	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	5	2	5	3	3	1

Bewertung der Moden der Teststruktur:

Unterbewertungen:

- Messbarkeit
- Eignung

```
eval_all := | for j ∈ 1..N_mode
            |   eval ← 5  if eval_freq[1,j] = 5 ∨ eval_suit[1,j] = 5
            |   eval ← (eval_freq[1,j] + eval_suit[1,j]) / 2  otherwise
            |   eval_all[1,j] ← | rund(eval)  if eval ≤ 5
            |                   | 5  otherwise
            |   eval_all
            | eval_all
```

evaluation_all_modes := stapeln(eval_mess, eval_suit, eval_all)

evaluation_all_modes =

	1	2	3	4	5	6	7	8	9	10
1	1	2	2	3	3	5	3	5	5	5
2	2	2	2	2	5	2	5	3	3	1
3	2	2	2	3	5	3	5	5	5	5

Noten der Eigenmoden

eval_all =

	1	2	3	4	5	6	7	8	9	10
1	2	2	2	3	5	3	5	5	5	5

Für eine gute Eignung für mehrere Parameter sollten deren Noten in gleichen Moden möglichst weit auseinander liegen.

```
eval_suit := | for j ∈ 1..N_mode
             |   eval ← 5 − |eval_p1[1,j] − eval_p2[1,j]|
             |   eval_suit[1,j] ← | rund(eval)  if eval ≤ 5
             |                    | 5  otherwise
             |   eval_suit
             | eval_suit
```

evaluation_suitability := stapeln(eval_p1, eval_p2, eval_suit)

evaluation_suitability =

	1	2	3	4	5	6	7	8	9	10
1	5	5	5	2	2	5	2	3	3	5
2	2	2	2	5	2	2	2	1	1	1
3	2	2	2	2	5	2	5	3	3	1

Noten der Eignung

eval_suit =

	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	5	2	5	3	3	1

Bewertung der Moden der Teststruktur:

Unterbewertungen:

- Messbarkeit
- Eignung

```
eval_all := | for j ∈ 1..N_mode
            |   eval ← 5  if eval_freq[1,j] = 5 ∨ eval_suit[1,j] = 5
            |   eval ← (eval_freq[1,j] + eval_suit[1,j]) / 2  otherwise
            |   eval_all[1,j] ← | rund(eval)  if eval ≤ 5
            |                   | 5  otherwise
            |   eval_all
            | eval_all
```

evaluation_all_modes := stapeln(eval_mess, eval_suit, eval_all)

evaluation_all_modes =

	1	2	3	4	5	6	7	8	9	10
1	1	2	2	3	3	5	3	5	5	5
2	2	2	2	2	5	2	5	3	3	1
3	2	2	2	3	5	3	5	5	5	5

Noten der Eigenmoden

eval_all =

	1	2	3	4	5	6	7	8	9	10
1	2	2	2	3	5	3	5	5	5	5

Gesamtnote:

Unterbewertungen:

- Gesamtbewertung der Moden

Anzahl der benötigten "guten" Moden (Note besser als 5):

n_soll := 2

```
eval_ts := | n_good ← 0
           | for k ∈ 1 .. N_mode
           |   n_good ← n_good + 1  if eval_all_{1,k} < 5
           | if n_good ≥ n_soll
           |   | eval ← sort(eval_all^T)
           |   | eval_ts ← mittelwert(submatrix(eval, 1, n_soll, 1, 1))
           |   | eval_ts
           | eval_ts ← 5  otherwise
           | eval_ts
```

Anzahl der "guten" Moden ermitteln

Berechnung des Durchschnitts der minimal notwendigen "guten" Noten

$sort(eval\_all^T)^T =$

	1	2	3	4	5	6	7	8	9	10
1	2	2	2	3	3	5	5	5	5	5

Gesamtnote der Teststruktur:

eval_ts = 2

Anhang G

Geometriemodifikation mit EA am Beispiel

Die Geometriemodifikation durch einen Evolutionären Algorithmus wird am Beispiel einer viertelsymmetrischen Struktur nachfolgend demonstriert. Das Konzept entspricht dem in Abschnitt 3.3.3. Die Bewertung erfolgt nach ihrer ersten Eigenfrequenz, welche durch eine ANSYS-Simulation ermittelt wird, und ihren lateralen Abmessungen. Optimierungsziel ist eine Struktur, welche eine festgelegte Eigenfrequenz bei einer bestimmten Maximalgröße besitzt.

Im folgenden Abschnitt werden Geometrie und Randbedingungen der Struktur und der Ablauf des Algorithmus erläutert, die Simulationsbedingungen werden anschließend aufgelistet. Abschließend erfolgt die Auswertung der Ergebnisse und in einem Fazit werden Verbesserungsvorschläge gemacht.

G.1 Aufbau

Die Geometrie der viertelsymmetrischen Demonstrationsstruktur ist in Abbildung G.1 dargestellt. Sie ist überbemaßt, da zusätzliche Parameter für die Gültigkeitsprüfung des Gens benötigt werden. Zusätzlich sind die Punkte und Flächen eingezeichnet, welche zu Erstellung eines vollparametrisierten ANSYS-Modells notwendig sind. Die notwendigen ANSYS-Makros für die Geometrie, die Randbedingungen und Modalanalyse sind bereitzustellen, die Parameterdatei wird durch den Evolutionären Algorithmus generiert. Die Dicke der Struktur ist auf $50\mu m$ festgelegt.

Die feste Einspannung wird bei $\pm$`ly` festgesetzt. Die laterale Ausdehnung in x-Richtung wird mit `mx_max`, in y-Richtung mit `my_max` bezeichnet. Die Maße `g1` und `g2` werden für die Gültigkeitsprüfung eines Gens benötigt.

Die Randbedingungen

$$\begin{aligned} \texttt{mx} + \texttt{lx} &\leq \texttt{mx\_max} \\ \texttt{my} + \texttt{g2} &\leq \texttt{my\_max} \\ \texttt{ly} &\leq \texttt{my\_max} \\ \texttt{my} &< \text{ly} \end{aligned}$$

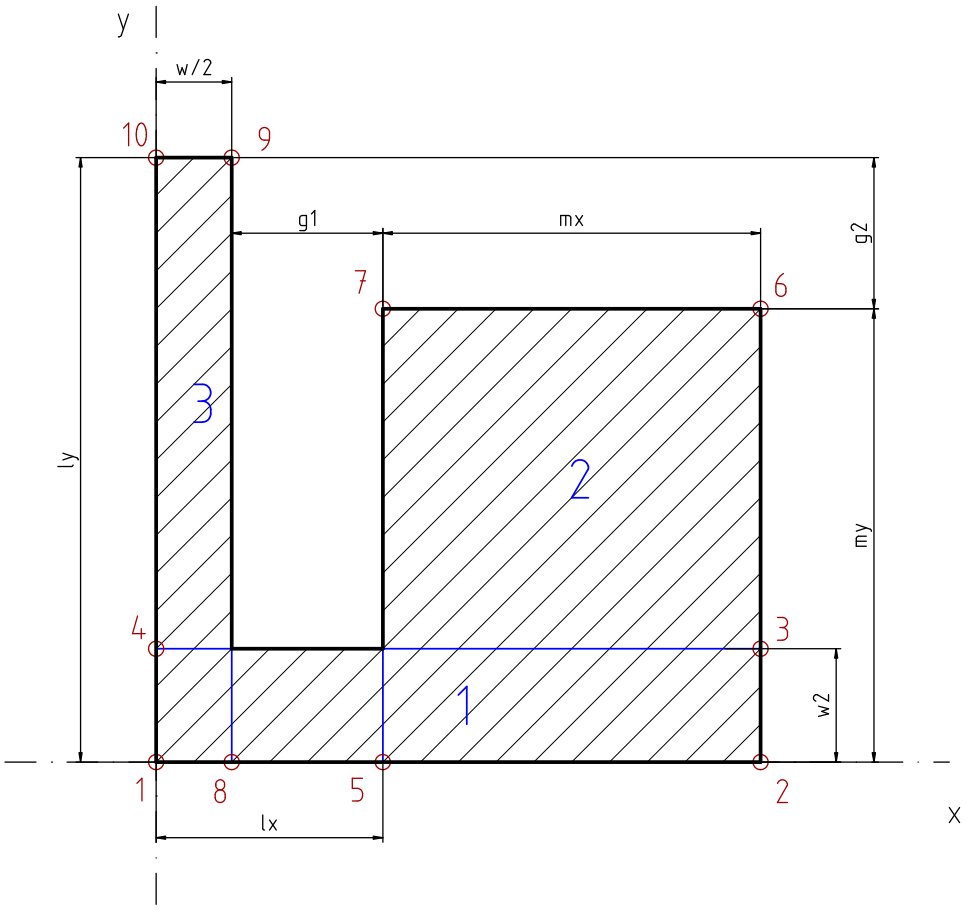


Abbildung G.1: Geometrie der Struktur s02

legen die Gültigkeit eines Gens fest. Sind sie nicht erfüllt, so ist das Gen nicht gültig. Das genetische Repräsentation (Chromosom) besteht damit aus den Parametern `mx my lx ly w2`. Diese müssen innerhalb festzulegender Intervalle liegen, sonst ist das Chromosom ebenso ungültig. In einer Textdatei werden der jeweilige Parametername, seine Intervallober- und -untergrenze leerzeichengetrennt zeilenweise hinterlegt (siehe Listing G.1).

Listing G.1: Genomdatei der Demonstrationsstruktur

```
MX 20 100
MY 20 100
LX 10 50
LY 20 125
W2 5 45
```

Die Optimierungsziele sind in Tabelle G.1 zusammengefasst.

Die Umsetzung des Evolutionären Algorithmus erfolgt in C++ unter Nutzung der Bibliothek GAlib (siehe Abschnitt 2.8.5). Zur Anwendung kommt ein Steady-State-GA mit einer Populationsgröße von 20 Individuen, welcher 10 Generationen

Tabelle G.1: Optimierungsziele

Parameter	Zielwert
1. Eigenfrequenz f_1, soll	=5000 Hz
`mx_max`	$\leq 250\mu m$
`my_max`	$\leq 250\mu m$
`g1`	$\geq 20\ \mu m$
`g2`	$\geq 20\ \mu m$

simulieren soll. Die Initialisierungsfunktion für Chromosome und die Bewertungsfunktion wurden angepasst. Für alle restlichen Einstellungen gelten die Standards der Bibliothek verwendet.

Die Initialisierung erzeugt einen Genotyp, bei welchem die Gene zufällige Werte aus ihrem entsprechend zulässigen Intervall zugewiesen bekommen. Die untere Grenze wird dabei stets durch das Intervall vorgegeben, die obere kann durch den verbleibenden Rest begrenzt werden, wenn die Randbedingungen erfüllt sind. Sie verschiebt sich damit nach unten.

Die Bewertung orientiert sich ebenso an den Randbedingungen. Sind diese nicht erfüllt, oder liegen die Gene außerhalb ihrer zulässigen Intervalle, wird der Genotyp mit 0 bewertet. Eine ANSYS-Simulation findet dann nicht statt, da unter Umständen schon der Modellaufbau fehlschlagen würde. Falls der Genotyp nicht ausgesondert wurde, erfolgt die Simulation der Struktur mit ANSYS. Der Genotyp wird dazu in der Parameterdatei des Modells gespeichert. Über das ANSYS-Simulationsmakro wird ein Bild des ersten Eigenmodes und der zugehörigen Eigenfrequenz abgelegt. Diese wird nachfolgend wieder in das C++-Programm eingelesen. Dieses ermittelt die Abweichung von der Zielfrequenz nach Gleichung G.1 und G.2 und bewertet den Genotyp nach Gleichung G.3 mit einer Punktzahl s.

$$\Delta f_1 = |f_{1,\text{soll}} - f_{1,\text{ist}}| \tag{G.1}$$

$$\Delta f_{1,\text{rel}} = \frac{|f_{1,\text{soll}} - f_{1,\text{ist}}|}{f_{1,\text{soll}}} \tag{G.2}$$

$$s = \begin{cases} 1 + {}^1/_{\Delta f_{1,\text{rel}}^2} & \text{, wenn} \Delta f_1 > 0.05 \\ 1000 & \text{, sonst} \end{cases} \tag{G.3}$$

mit
- $f_{1,\text{soll}}$ ... Sollwert der ersten Eigenfrequenz
- $f_{1,\text{ist}}$ ... erste Eigenfrequenz eines Individuums
- Δf_1 ... absolute Abweichung vom Sollwert der Eigenfrequenz
- $\Delta f_{1,\text{rel}}$ ... relative Abweichung vom Sollwert der Eigenfrequenz
- s ... Punktzahl eines Individuums

Die Punktzahl s ist in diesem Fall gleich der Fitness eines Individuums. Diese ist umso besser, je näher ihre Eigenfrequenz an der Zielfrequenz liegt. Sie ist, wie bereits erwähnt, Null, wenn das Individuum die zulässige Maximalgröße überschreitet.

G.2 Simulationsbedingungen

Die Simulation wurde auf dem Referenzserver `pandora` (pandora.hrz.tu-chemnitz.de) durchgeführt. Die Spezifikation des Servers sind nachfolgend aufgeführt.

Prozessor	Quad-Core AMD Opteron^{TM}Processor 8356
Taktfrequenz	2133 MHz
Datenbreite	32 Bit
Arbeitsspeicher	2 GB
Betriebssystem	Scientific Linux 5

Die Version der verwendeten Programme und Bibliotheken finden sich in Tabelle G.2.

Tabelle G.2: Versionen verwendeter Programme und Bibliotheken

Programm / Bibliothek	Version
ANSYS	11.0 (Build 2007.0830)
GAlib	2.4.7

G.3 Auswertung

Die Simulation des Evolutionären Algorithmus wurde mit einer Startpopulation von 20 Individuen durchgeführt. In Tabelle G.3 sind Simulationsparameter und Zeitdaten zusammengefasst. Die Anzahl der Generationen wurde auf 10 begrenzt, damit wird zwar der Zielwert unter Umständen nicht erreicht, aber es ist eine Tendenz feststellbar. Die Gesamtdauer der Simulation betrug rund 12 Minuten, wobei über 80% für die Modalanalyse mit ANSYS abfielen — nicht mitgerechnet die Ladezeit von ANSYS. Damit ist die FE-Analyse der zeitkritische Punkt.

Tabelle G.3: Werte des simulierten EA

Eigenschaft	Wert
Anzahl der Generationen	10
Individuen in der Startpopulation	20
Gesamtanzahl der simulierten Individuen	111
davon einzigartige	96
Durchschnittliche Simulationsdauer einer Generation	69 s
Gesamtdauer der Simulation	766 s
davon ANSYS-Simulation	83,9%

Tabelle G.4 vergleicht die erreichten Werte mit den Zielwerten. Die maximalen Abmessungen werden eingehalten, die erste Eigenfrequenz kommt der Zielfrequenz nahe. Deutlicher wird dies in Abbildung G.2.

Tabelle G.4: Vergleich von Soll- und Istwerten

Eigenschaft	Sollwert	erreichter Wert	Ziel erreicht
1. Eigenfrequenz [Hz]	5000	5680.1	annähernd
max. Abmessung in X [μm]	500	297.2	✓
max. Abmessung in Y [μm]	500	67.8	✓

Die mittlere erste Eigenfrequenz jeder Generation nähert sich mit steigender Generation an die Solleigenfrequenz an. Ausreißer sind dabei weniger relevant. Die besten Individuen jeder Generation weisen wesentlich kleinere Abweichungen auf. Die global beste Frequenz wird in drei Generationen erreicht.

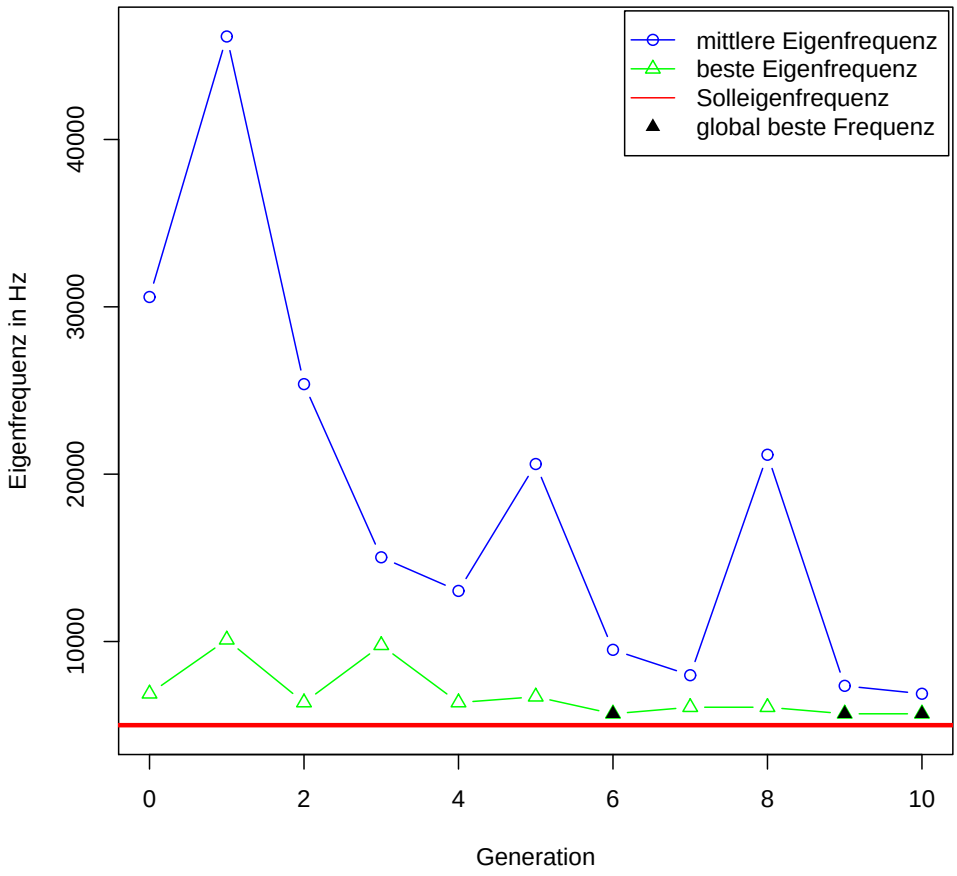
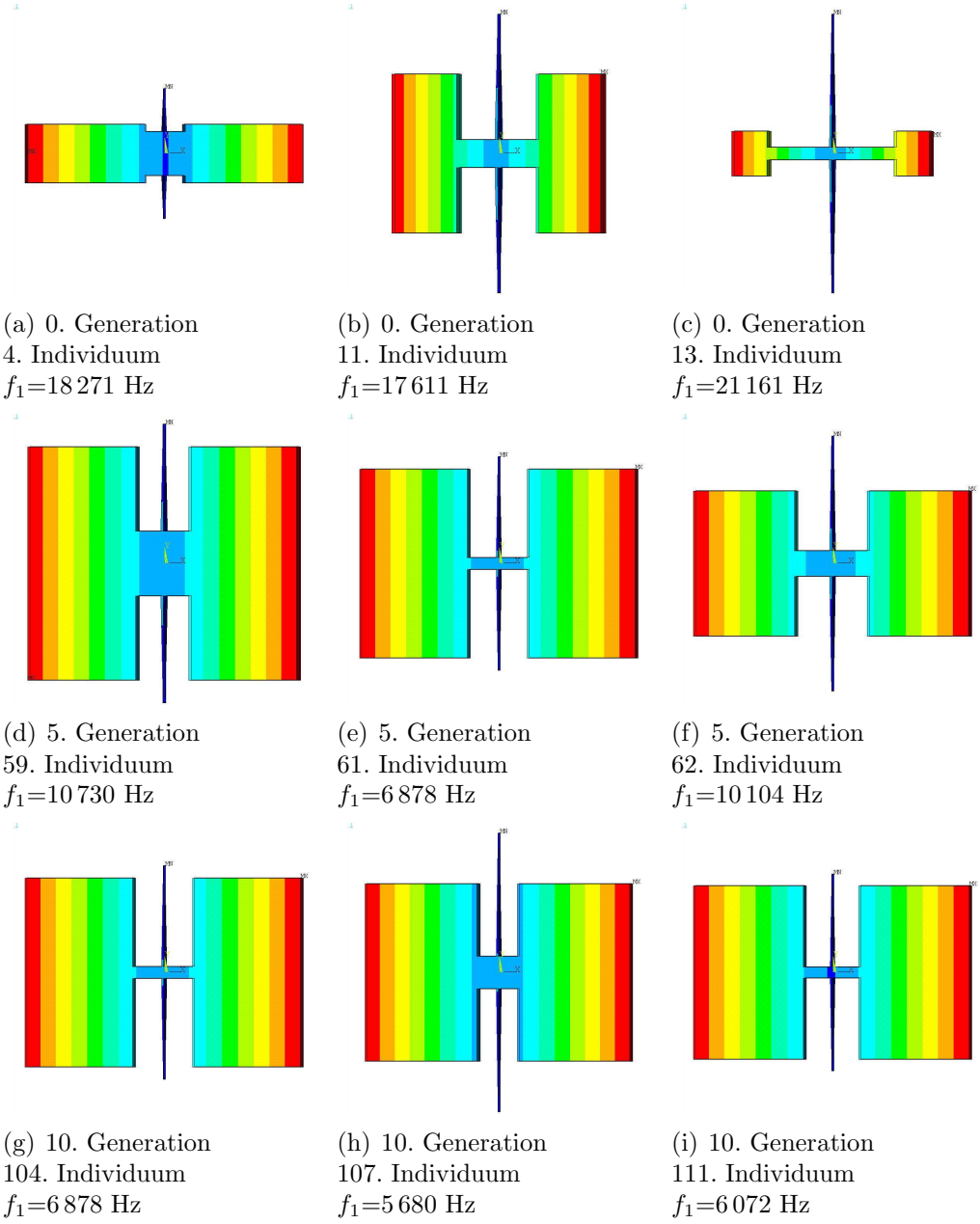


Abbildung G.2: Eigenfrequenzentwicklung über die Generation

Abbildung G.3 zeigt einige Individuen aus drei verschiedenen Generationen mit ihrer ersten Eigenfrequenz. Ist die Geometrie in der Startpopulation 0 sehr unterschiedlich, so nähert sie sich in den folgenden Generationen immer weiter an eine an, welche die Zielparameter erfüllt.



(a) 0. Generation
4. Individuum
f_1=18 271 Hz

(b) 0. Generation
11. Individuum
f_1=17 611 Hz

(c) 0. Generation
13. Individuum
f_1=21 161 Hz

(d) 5. Generation
59. Individuum
f_1=10 730 Hz

(e) 5. Generation
61. Individuum
f_1=6 878 Hz

(f) 5. Generation
62. Individuum
f_1=10 104 Hz

(g) 10. Generation
104. Individuum
f_1=6 878 Hz

(h) 10. Generation
107. Individuum
f_1=5 680 Hz

(i) 10. Generation
111. Individuum
f_1=6 072 Hz

Abbildung G.3: Ausgewählte Individuen aus verschiedenen Generationen

G.4 Fazit

Die Demonstration an einer Beispielstruktur zeigt, dass eine Geometrie mit Hilfe eines Evolutionären Algorithmus dahingehend optimiert werden kann, dass die

resultieren Struktur vorgegebene Randbedingungen einhält beziehungsweise diesen nahe kommt. Wie genau diese eingehalten werden, hängt von der festgelegten Bewertung beziehungsweise Fitfunktion ab. Der Punktwert und damit die Fitness für ein Individuum muss allgemein umso größer sein, je besser das Optimierungsziel getroffen wird.

Der Einsatz des Steady-State-Algorithmus zeigt das gewünschte Verhalten. Der Simple-Algorithmus ist weniger gut geeignet, da sich jede Generation aus neuen Individuen zusammensetzt. Gute bleiben damit nicht erhalten (siehe Abschnitt 2.8.5).

Für den verwendeten Algorithmus sind weitere Verfeinerungen möglich. Beispielsweise können die Mutations- und die Kreuzungsfunktion angepasst werden. Denkbar wäre eine gutartige Mutation, welche nur gültige Gene hervorbringt. Weiterhin können die Kreuzungs- und Mutationsraten, sowie der Anteil der die Generationen überlebenden Individuen verfeinert werden.

Da die ANSYS-Simulation jedes Individuums der zeitkritische Faktor ist, muss diese für größere Simulationen optimiert werden. Beeinflusst wird dies wesentlich von der verwendeten Hardware. Um die Anzahl der Simulationen zu reduzieren, wäre eine Datenbank günstig, welche bereits simulierte Individuen mit ihrer Fitness enthält. Tritt dann ein bereits bewertetes auf, können die Daten verwendet werden.

Literaturverzeichnis

[ABL+99] Ayón, A., R. Braff, C. Lin, H. Sawin und M.A. Schmidt: *Characterization of a time multiplexed inductively coupled plasma etcher.* Journal of The Electrochemical Society, 1999.

[AM05] Asgary, R. und K. Mohammadi: *Using fuzzy probabilistic neural network for fault detection in MEMS.* In: Mohammadi, K. (Herausgeber): *Intelligent Systems Design and Applications, 2005. ISDA '05. Proceedings. 5th International Conference on*, Seiten 136–140, 2005.

[ANS] *Release 11.0 Documentation for ANSYS.*

[BHY+02] Bae, S. Y., K. J. Hayworth, K. Y. Yee, K. Shcheglov und D. V. Wiberg: *High performance MEMS micro-gyroscope.* Proceedings of SPIE - The International Society for Optical Engineering, 4755:316–324, 2002. Cited By (since 1996): 9.

[Boo] *Boost 1.38.0 Library Documentation.* `http://http://www.boost.org/`, 22.03.2009.

[Che93] Chester, Michael: *Neural networks.* Prentice Hall, 1993.

[CLV07] Coello Coello, Carlos A., Gary B. Lamont und David A. Van Veldhuizen: *Evolutionary algorithms for solving multi-objective problems.* Genetic and evolutionary computation series. Springer, 2. Auflage, 2007.

[CM07] Chen, B. und J. Miao: *Influence of deep RIE tolerances on comb-drive actuator performance.* Journal of Physics D: Applied Physics, 40(4):970–976, 2007.

[DeJ06] DeJong, Kenneth Alan: *Evolutionary computation.* A Bradford book. MIT Press, 2006.

[FGW+04] Fan, Zhun, E.D. Goodman, Jiachuan Wang, R. Rosenberg, Kisung Seo und Jianjun Hu: *Hierarchical evolutionary synthesis of MEMS.* In: Goodman, E.D. (Herausgeber): *Evolutionary Computation, 2004. CEC2004. Congress on*, Band 2, Seiten 2320–2327 Vol.2, 2004.

[GD06] Gerlach, Gerald und Wolfram Dötzel: *Einführung in die Mikrosystemtechnik.* Fachbuchverlag Leipzig im Carl Hanser Verlag, 2006.

[GM98] Groth, Clemens und Günter Müller: *FEM für Praktiker - Band 3: Temperaturfelder*, Band 3 der Reihe *FEM für Praktiker*. expert-Verlag, 1998.

[GOS96] Gupta, R.K., P.M. Osterberg und S.D. Senturia: *Material property measurements of micromechanical polysilicon beams.* Proc. SPIE 1996: Microlithograph and Metrology in Micromachining II, Seiten 39–45, 1996.

[Gup00] Gupta, R.K.: *Electronically probed measurements of MEMS geometries.* Journal of Microelectromechanical Systems, 9(3):380–389, 2000.

[HA97] Hubbard, T.J. und E.K. Antonsson: *Cellular automata modeling in MEMS design.* Sensors and Materials, 9(7):437–448, 1997.

[Hil04] Hiller, Dr.-Ing. Karla: *Technologieentwicklung für kapazitive Sensoren mit bewegten Komponenten.* Doktorarbeit, Technische Universität Chemnitz, 2004. Habilitationsschrift.

[HKB+05] Hiller, K., M. Küchler, D. Billep, B. Schröter, M. Dienel, D. Scheibner und T. Gessner: *Bonding and Deep RIE - A powerful combination for high aspect ratio sensors and actuators.* In: *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, Band 5715, Seiten 80–91, Fraunhofer Institute for Reliability and Microintegration, Department Micro Devices and Equipment, Chemnitz, Germany, 2005.

[IT07] Ilumoka, A. und Hong Lang Tan: *MEMS Failure Probability Prediction and Quality Enhancement Using Neural Networks.* In: Tan, Hong Lang (Herausgeber): *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, Seiten 322–326, 2007.

[JJQH05] Jensen, S., J.M. Jensen, U.J. Quaade und O. Hansen: *Uniformity-improving dummy structures for Deep Reactive Ion Etching (DRIE) processes.* In: *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, Band 5715, Seiten 39–46, Department of Physics, Interdisciplinary Research Center for Catalysis (ICAT), Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, 2005.

[Kle00] Klein, Bernd: *FEM.* Vieweg, 2000.

[KW99] Knothe, Klaus und Heribert Wessels: *Finite Elemente - Eine Einführung für Ingenieure.* Springer, 1999.

[LA98] Li, H. und E.K. Antonsson: *Evolutionary Techniques in MEMS Synthesis.* 25th Biennial Mechanisms Conf., ASME Design Engineering Technical Conf., (DETC'98), 1998.

[LAZ04] Litovski, V., M. Andrejevic und M. Zwolinski: *ANN based modeling, testing and diagnosis of MEMS [capacitive pressure transducer*

example]. In: Andrejevic, M. (Herausgeber): *Neural Network Applications in Electrical Engineering, 2004. NEUREL 2004. 2004 7th Seminar on*, Seiten 183–188, 2004.

[Lea06] Lea, L.: *DRIE Etched Silicon MEMS*. MEMS Manufacturing, Seiten 19–22, 2006. Cited By (since 1996): 1.

[LTC+03] Logeeswaran, V. J., F. E. H. Tay, M. L. Chan, F. S. Chau und Y. C. Liang: *First Harmonic (2f) Characterisation of Resonant Frequency and Q-Factor of Micromechanical Transducers*. Analog Integrated Circuits and Signal Processing, 37:14–33, 2003. Cited By (since 1996): 1.

[Maz92] Mazzetti, Alessandro: *Praktische Einführung in neuronale Netze*. Heise, 1992.

[Mei07] Meinig, Marco: *Verfahren zum Test von MEMS-Bauelementen durch die Messung der Eigenfrequenzen*. Diplomarbeit, Technische Universität Chemnitz, 2007.

[MH07] Martin, Ken und Bill Hoffmann: *Mastering CMake - a cross-plattform build system*. Kitware, 2007.

[MK07] Musciano, Chuck und Bill Kennedy: *HTML & XHTML - the definite guide*. O'Reilly, 2007.

[NEW] *Documentation for newmat11, a matrix library in C++*. `http://www.robertnz.net/nm11.htm`, 22.03.2009.

[Nis03] Nissen, Steffen: *Implementation of a Fast Artificial Neural Network Library (FANN)*, Oktober 2003.

[Nis05] Nissen, Steffen: *Neural Networks Made Simple*. Software2.0, 1:6, Februar 2005.

[Nis09] Nissen, Steffen: *Fast Artificial Neural Network Library*, 01 2009.

[Nor03] *Computational Intelligence*, Februar 2003. VDE/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA).

[Opt] *Dokumentation des Optimierungsframeworks für Teststrukturen*.

[OS97] Osterberg, P.M. und S.D. Senturia: *M-test: A test chip for MEMS material property measurement using electrostatically actuated test structures*. Journal of Microelectromechanical Systems, 6(2):107–118, 1997.

[PA98] Petersan, P. J. und S. M. Anlage: *Measurement of resonant frequency and quality factor of microwave resonators: Comparison of methods*. Journal of Applied Physics, 84(6):3392–3402, 1998. Cited By (since 1996): 40.

[Pola] *Polytec Scanning Vibrometer - Theory Manual.*

[Polb] *Homepage Polytec.* `http://www.polytec.com/`, 22.03.2009.

[SFS+08] Shaporin, Alexey, Roman Forke, Ralf Schmiedel, Wolfram Dötzel, Jan Mehner, Detlef Billep und Thomas Gessner: *Test-Structures for Wafer Level Microsystems Characterization.* In: *Smart Systems Integration 2008*, Seiten 528–531, 2008.

[SGM06] Stelzmann, Ulrich, Clemens Groth und Günther Müller: *FEM für Praktiker - Band 2: Strukturdynamik*, Band 2 der Reihe *FEM für Praktiker.* expert-Verlag, 4. Auflage, 2006.

[Sha05] Shaporin, A.: *Efficient characterization method for MEMS devices.* In: *Electron Devices and Materials, 2005. Proceedings. 6th Annual. 2005 International Siberian Workshop and Tutorials on*, Seiten 21–26, 2005.

[SHD05] Shaporin, A., M. Hanf und W. Doetzel: *Novel characterization method for MEMS devices.* In: *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, Band 5716, Seiten 198–206, Chemnitz University of Technology, Faculty of Electrical Engineering and Information Technology, Department of Microsystems and Precision Engineering, Reichenhainer Str. 70, Chemnitz, D-09126, Germany, 2005.

[SHF94] Schöneburg, Eberhard, Frank Heinzmann und Sven Feddersen: *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley, 1. Auflage, 1994.

[SHF+07] Shaporin, A., M. Hanf, R. Forke, J. Mehner und W. Gessner, T.and Dötzel: *Test-Structure based MEMS Characterization Technique.* In: *Smart Systems Integration*, Seiten 625–627, 2007.

[SHG90] Schöneburg, Eberhard, Nikolaus Hansen und Andreas Gawelczyk: *Neuronale Netzwerke.* Markt-u.-Technik-Verlag, 1990.

[SMR+04] Scheibner, D., J. Mehner, D. Reuter, U. Kotarsky, T. Gessner und W. Dötzel: *Characterization and self-test of electrostatically tunable resonators for frequency selective vibration measurements.* Sensors and Actuators, A: Physical, 111(1):93–99, 2004. Cited By (since 1996): 6.

[SSS+09] Shaporin, Alexey, Petra Streit, Hendrik Specht, Jan Mehner und Wolfram Dötzel: *Novel test structures for characterization of microsystems parameters at waferlevel.* Band 7206, Seite 72060E. SPIE, 2009.

[Str03] Stroustrup, Bjarne: *Die C++ Programmiersprache.* Programmers choice. Addison-Wesley, 4th Auflage, 2003.

[Str07] Streit, Petra: *Entwicklung eines Tools für ANSYS zur Modifizierung von 3D-Volumenmodellen*, 2007.

[ÜKP05] Überhuber, Christoph W., Stefan Katzenbeisser und Dirk Praetorius: *MATLAB 7 - Eine Einführung*. Springer, 2005.

[Wal96] Wall, Matthew: *GAlib: A C++ Library of Genetic Algorithm Components*, 1996.

[WJ03] Welch, Brent B. und Ken Jones: *Practical programming in Tcl/Tk*. Prentice Hall PTR, 4. Auflage, 2003.

[ZAKS06] Zhang, Y., A.M. Agogino, R. Kamalian und C.H. Sequin: *Design synthesis of microelectromechanical systems using genetic algorithms with component-based genotype representation*. In: *8th Annual Genetic and Evolutionary Computation Conference 2006*, Band 1, Seiten 731–738, Seattle, WA, 2006.

[ZKAS05] Zhang, Y., R. Kamalian, A.M. Agogino und C.H. Se?quin: *Hierarchical MEMS synthesis and optimization*. In: V.K., Varadan (Herausgeber): *Smart Structures and Materials 2005 - Smart Electronics, MEMS,, BioMEMS, and Nanotechnology*, Band 5763, Seiten 96–106, San Diego, CA, 2005.

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Chemnitz, den 4. April 2009

Petra Streit